DOCUMENT RESUME

ED 233 696 IR 010 786

AUTHOR	Watson, Nancy Ralph, Ed.
TITLE	Microcomputers in Education: Uses for the '80s. Proceedings of the Annual Microcomputer Conference (2nd, Tempe, Arizona, January 15-16, 1982). Publication No. 3.
INSTITUTION	Arizona State Univ., Tempe. Dept. of Elementary Education.
PUB DATE	82
NOTE	296p.
AVAILABLE FROM	Microcomputer Research Clinic, Arizona State University, Payne 303, College of Education, Tempe, AZ 85287 ($20.00 per copy).
PUB TYPE	Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - Descriptive (141)
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Literacy; *Computer Programs; Educational Administration; Educational Media; Educational Research; Elementary Secondary Education; Inservice Teacher Education; Mathematics Instruction; *Microcomputers; Parent Teacher Cooperation; Physical Disabilities; Postsecondary Education; *Programing; School Psychologists; Science Instruction
IDENTIFIERS	Apple II

ABSTRACT

The 30 conference papers in this collection are presented in 6 categories. Five overviews discuss innovative uses of computers in education (Dorothy K. Deringer); microcomputers in instructional research (Alan M. Lesgold); microcomputers in the schools (Mitchell Batoff, Gary G. Bitter); and the courseware crisis (Barbara R. Sadowski). Research and projects described focus on basic skills through microcomputers (Kay Gilliland); data acquisition and handling using microcomputers (Pauline Fortier Briggs); learning logic and geometry with microcomputers (Teri Hoch Perl and Ann M. Piestrup); Young People's LOGO Association (James H. Muller); teacher training for using microcomputers in science education (Malcolm Wells and Gary G. Bitter); and using Apple microcomputers with young handicapped children (Laura F. Meyers and Terri Rosegrant). Discussions of administrative uses focus on administrators and computers (James P. Boyle); administrative applications (Patrick J. Carr); equipment acquisition (Tom Keller); microcomputers in school psychology (Marley J. Watkins); microcomputers in the classroom and at home (Pat Kuhn); teacher training workshops (John Losse and Rita Richards); and parental involvement (Greg Wojtulewicz). Papers on computer assisted instruction (CAI) describe using microcomputers in chemistry and physics instruction (David Byrum); creating CAI materials with Apple Pilot (Ruth A. Camuse); designing courseware in K-8 mathematics (James H. Wiebe); making software interesting (Jeffrey and Karen Raynor); and mathematical problem solving with microcomputers (J. F. Weaver). Computer literacy is discussed as content (Janis Rosenblum and Shirley Frye); a computer literacy project is described (Bruce Eldredge); and the importance of computers to education is considered. Presentations on programming include an outline of BASIC (Marilyn Sue Ford); good programming techniques (Homer Baker and Raymond Preston); tips for programming (Richard Dixon); and a programming project for high school students (Robert Zenor). A glossary and listing of microcomputer journals complete the proceedings. (ESR)

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

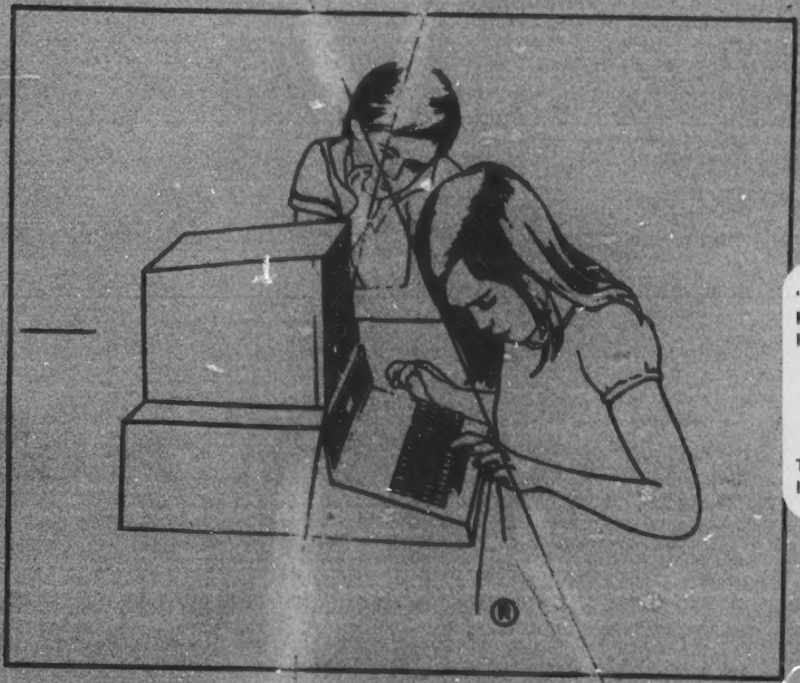
- Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

ED233696

COOPERATIVE OF FACULTY FOR TEACHING, RESEARCH AND SERVICE IN ELEMENTARY MATH, SCIENCE AND SOCIAL STUDIES

MICROCOMPUTERS IN EDUCATION
Uses for the 80's

Proceedings of the
Second Annual Microcomputer Conference
Arizona State University
Tempe, Arizona 85287



"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY
Gary Bitter

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR010786

Publication No. 3
1982

DEPARTMENT OF ELEMENTARY EDUCATION • PAYNE HALL ROOM B-225
ARIZONA STATE UNIVERSITY • TEMPE, ARIZONA 85287 • (602) 965-3711

MICROCOMPUTERS IN EDUCATION
Uses for the '80s

CONFERENCE PROCEEDINGS

EDITOR

Nancy Ralph Watson
Arizona State University

•

CONFERENCE DIRECTORS

Nancy Ralph Watson
Arizona State University

Gary. G. Bitter
Arizona State University

Conference held at Arizona State University
January 15th and 16th, 1982

IR010786

TABLE OF CONTENTS

SUMMARY OF CONFERENCE....................iv

I. OVERVIEWS....................1

The New Federalism: A Do-It-Yourself Approach To Computers-in-Education....................2
Dorothy K. Deringer

Microcomputers In Instructional Research....................30
Alan M. Lesgold

Why and Where Should Microcomputers be placed in Schools....................48
Mitchell Batoff

Microcomputers Go To School....................61
Gary G. Bitter

One Solution to the Courseware Crisis....................70
Barbara R. Sadowski

II. RESEARCH AND PROJECTS....................75

Basic Skills Through Microcomputers....................76
Kay Gilliland

Data Acquisition, Data Handling Considerations Using Microcomputers....................83
Pauline Fortier Briggs

✓ Early Learning of Logic and Geometry Using Microcomputers....................86
Teri Hoch Perl and Ann M. Piestrup

Here Come The Turtles!....................87
James H. Muller

Training Teachers for the Effective Use of Microcomputers in Science Education....................91
Malcolm Wells and Gary G. Bitter

The Use of Apple Computers with Young Handicapped Children....................105
Laura F. Meyers and Terri Rosegrant

III. ADMINISTRATIVE USES...107

Administrators and Computers - Now and Tomorrow........108
James P. Boyle

Administrative Applications
A Frame of Reference..............................114
Patrick J. Carr

Computer Equipment Acquisition Guide for the
Educational Administrator.........................126
Tom Keller

Microcomputers in the Practice of
School Psychology.................................131
Marley J. Watkins

Microcomputers - Tools in the Classroom-
Tutors at Home....................................140
Pat Kuhn

A Model for Teacher Training Workshops................150
John Losse and Rita Richards

Parental Involvement: The Key to Success..............153
Greg Wojtulewicz

IV. COMPUTER ASSISTED INSTRUCTION.....................................161

Chemistry, Physics and Microcomputers.................162
David Byrum

Creating Computer Assisted Instruction with
Apple Pilot.......................................167
Ruth A. Camuse

Designing Microcomputer Software in K-8
Mathematics.......................................191
James H. Wiebe

Educational Software Doesn't Have
To Be Boring......................................206
Jeffrey Raynor and Karen Raynor

Mathematical Problem Solving and Microcomputers:
One Way to Use them Judiciously...................209
J. F. Weaver

V. LITERACY....219

Computer Literacy as Content....220
Janis Rosenblum and Shirley Frye

Computer Literacy Project....225
Bruce Eldredge

The Importance of Computers to Education....231

VI. PROGRAMMING....235

BASIC in Ten Easy Steps....236
Marilyn Sue Ford

Don't Let Your Programs Bug You....245
Homer Baker and Raymond Preston

Home-Brewed Software....275
Richard Dixon

Programming for High School Students....282
Robert Zenor

VII. GLOSSARY and JOURNALS....283

The material presented in this publication does not imply the expression of any opinion on the part of Arizona State University, the Department of Elementary Education, or associates thereof.

For further information regarding specific papers included herein, write directly to the address provided with each article.

SUMMARY OF CONFERENCE

Approximately 700 educators, administrators, students and parents from all over the country attended the Second Annual Microcomputers in Education Conference at Arizona State University in Tempe, Arizona -- nearly double the number attending the first conference last winter. This reflects the tremendous interest in the pursuit of information to develop teaching competencies in the preparation for the instruction of curriculum relating to the area of microcomputers.

A total of 60 different sessions were offered in the areas of computer assisted instruction, programming, research, administrative uses and literacy. The papers included in this publication reflect the majority of those presented during the conference. In addition to the formal sessions, the facilitators in four "hands-on" rooms, representing Apple, Radio Shack, Commodore Pet and Texas Instruments, offered participants a variety of first-hand experiences on micros.

On the last day of the conference the results of a statewide K-8 programming contest were announced. This contest was sponsored by the Arizona Computer Users in Education organization, an affiliate of the Association for Educational Data Systems, Inc. The contest winners were introduced, along with their sponsoring teachers, and presented with gifts donated by local computer dealers. This ceremony served as a reminder that the efforts of teachers in this field may be reflected with this kind of recognition for students. The winners were each provided with the appropriate computer on which to demonstrate their programs to teachers, administrators, parents and friends.

With the response of the participants at this conference, as well as to the many others being held across the country, we anticipate the need for and encourage the sharing of materials and experiences. We urge you to continue attending conferences, supporting fellow educators in their publishing efforts and sharing your ideas so that we can learn from each other.

I. OVERVIEWS

THE NEW FEDERALISM: A Do-it-Yourself Approach to Computers in Education

Keynote Address

Microcomputer Conference
Arizona State University

by Dorothy K. Deringer
National Science Foundation
525 Wisconsin Avenue NW
Washington D.C. 20550

I am happy to be here in Arizona today. The title of this talk is "The New Federalism: A Do it Yourself R&D Approach to Computers in Education." R&D in computers in education is an area in which Arizona is several steps ahead of many other states in the Union. When my colleague at NSF Andrew Molnar returned from last year's conference at which he was the keynote speaker, he was very complimentary of the intensity of the conference participants in learning about all aspects of the new micro technogy and of Dr. Bitter and his colleagues, your microcomputer laboratory, your teacher programs and your other strong efforts to respond to the challenges and opportunities that computers in schools and universities, the workplace, and our homes offer us.

Computers--through their use in a variety of communication, information, and manufacturing systems--have the potential for being the most powerful intellectual tools that mankind has ever invented. The United States took an early lead in the design and manufacture of computers, and the U.S. is still the world leader in R&D, and in creating innovative educational uses. We are the world's largest manufacturers of computers and the world's largest consumers. Our high technology industries still enjoy a positive international balance of payments. The computer has also become the symbol for international power and prestige. Embargos of high technology products are frequently the result of diplomatic and political power plays between the United States and other governments.

Foreign students come from around the world to study in our graduate departments, more than making up in numbers for the loss of U.S. students to industrial employment. While 41% of all foreign graduate students enroll in engineering, math and computer science is next highest (30%) or 71% are exposed to computing curriculum in U.S. sciences. The United states still outshines all the rest of the world in the creation, production and use of the computer, the principal component in the high-growth knowledge-based industries of the future.

A key to maintaining our position, I believe, is a population educated in the use of computers in what ever their jobs may be--a computer-literate or computer-fluent populace. For many years, the National Science Foundation has had as part of its mandate the responsibility "to foster the use of computers for the improvement of education and research in the sciences."[1] Under that mandate we have undertaken a variety of innovative projects. The programs attracted highly creative people who pioneered many of the basic concepts used in communication.

(1) Back in the 60's we made a grant to John Kemeney at Dartmouth (later president of that institution) who produced a new programming language which was simpler and closer to normal English than its predecessors--it opened up computing to vastly greater numbers of people. The language was named BASIC.

(2) In the late 60's an award went to the University of Illinois to work on their idea for using the computer as an interactive learning device. They eventually named it PLATO.

(3) In the early 70's a first grant was made to a little known mathematician at MIT who believed children might find computers fascinating and learn better. They did and Seymour Papert was off and running...

(4) Finally, I mention also the intelligent videodisc projects and the publication "Technology in Science Education--The Next 10 Years."[2]

Last years NSF's Science and Engineering Education Directorate invested $17 million in computer-related activities. I'd like to share with you some of the activities we have been working on at NSF over the last two years, particularly in regard to developing prototypes for computers in mathematics instruction.

NSF Projects - Pre College Mathematics Using Computers

All of the 36 projects funded in FY 80 and 81 emphasize the development of prototypes of courseware for teaching, learning and doing mathematics. The projects suggest a series of questions that the community is pondering, and a number of areas for further research and development.

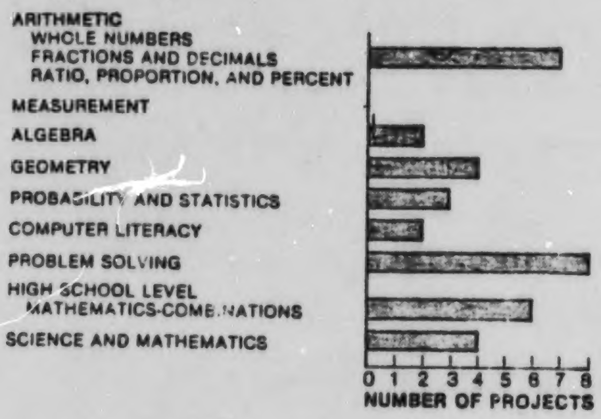
What topics in mathematics attract experimentation?

The first seven categories of Figure 1 are based on the categories of mathematics used in a report from the National Council of Teachers of Mathematics, "Priorities in School Mathematics (PRISM)".[3] The mathematical topics chosen by the projects reflect the high priorities of the school community. The large number of projects in arithmetic responds to the stress on basic skills. Measurement experssed in portions of one or two projects but is not the primary interest of any of them.

11

FIGURE 1

FY 1980 AND FY 1981
PRE-COLLEGE MATHEMATICS USING COMPUTERS
PROJECTS BY TOPIC



Eight projects are working on problem solving, and the National Council of Teachers of Mathematics Agenda for Action (Recommendation 1) stresses problem solving as one of the most important activities for mathematics education in the '80s. [4]

The category, high-school-level mathematics combinations, includes projects which are attempting to discover the most useful ways of incorporating computer graphics in trigonometry, geometry, algebra and even arithmetic. The projects are trying to weave the computer into the high school mathematics curriculum in places where students or teachers have special difficulties in teaching or learning mathematics.

Can young children learn mathematics, using computers?

Ten years ago the conventional wisdom was that a 4th grader was the youngest child that could use computers effectively. Microcomputers with color and sound have changed things a lot. Three projects (Piestrup, Kraus, Gattegno) are working with children in the first three grades.

William Kraus at Wittenberg University is working on a strategy for using the game of golf to teach estimation of angles and lengths. In the golf game, the lower righthand corner of the screen indicates the quadrants of the circle and the angles 0, 90, 180, and 270. The black line above the word "how" is the length of the units to be estimated for the distance of the drives and putts. The game can be played with up to four players. Both placement and numbers of sand traps and water hazards vary with each game. Placement of the tee and green is also varied.

Preliminary results indicate that very young children can become "mathematics-ready" and can learn mathematics by using computers.

Will interactive graphics improve learning? For whom?

Many of these projects are conducted in urban schools: they involve many minorities and are concerned with the problems women have experienced historically in learning mathematics. Adults are the focus of projects at Virginia Commonwealth University and the University of California at Santa Barbara; and adults will be part of the field-test populations of other projects.

The computer is a promising vehicle for making mathematics more accessible and understandable for a greater proportion of the population than succeeded in mathematics using just a teacher, textbook and the traditional classroom.

Can students improve their abilities to solve problems?

The term "problem solving" covers a wide variety of topics. The current interest in problem solving includes the mathemaatical statement or word problems of the past. It also includes decision-making skills and procedures and real-world problems in which there may be more than one correct answer to the problem. One of the goals is to help students to learn good methods for identifying, analyzing and solving problems. The computer can illustrate the situation with moving graphics or (if a videodisc is used) with still or moving pictures of the event. The student can explore and experiment by getting feedback from the computer before deciding which solution is appropriate. The problems can also have realistic numbers rather than numbers chosen to "come out even."

Why not use color?

Many projects are using both color and sound. Other than as attention-getting features, very little rationalization is given for why they are useful, but undoubtedly the relatively modest additional hardware investment required is one reason for choosing them. Many of the projects using black and white monitors chose them intentionally because they provide the better resolution which is necessary for accurate graphs and diagrams. Effective uses of color and sound are both areas of opportunity for further research and development activities. Only one of these projects, Wallace Feurzeig's at Bolt, Beranek and Newman, is researching the effective uses of sound in mathematics courseware. Good research on both the motivational and instructional reinforcement qualities of color and sound would be useful in creating more effective courseware. Public support for this research would assure that the findings were in the public domain.

Probably the most compelling use of color thus far is Sharon Dugdale's "Green Globs," an algebra game. The objective of the game is to write equations whose curves touch (and explode) all of the Green Globs. The more Globs touched by an equation, the higher the score.

The computer keeps a record of the highest scoring games, called the Hall of Fame, and other students can go back and replay these successful games. The students can learn from each other through this replay strategy as well as by working together on games.

How will the structure of the learning environment change?

Some people feel that the first way a computer enters the classroom is as a tool for the teacher. If all materials are completely individualized, it becomes more difficult for schools to use the computer effectively with a small, initial hardware purchase. Very few schools will initially buy computers for each two or three children. Some projects are writing courseware for the computer used as a teacher tool; others create materials for students to use individually or in groups.

The computer-centered classroom is a concept that is appearing more frequently in classrooms. Seymour Papert of M.I.T. described this approach as "the computer as pencil." Just as teachers now working with students in class use pencils and paper and perhaps a hand calculator, in the future each student will have his own computer. The computer is a student tool within a classroom for creating programs or doing algebra problems. This approach will not seem unreasonable when a powerful computer is as expensive as a hand calculator is today. A variety of new environments seems promising for our schools in the future.

What other conditions are necessary to realize these improvements?

Identifying diverse, innovative, promising and successfull new approaches is a crucial part of the challenge. Innovative strategies require the attention of the educational community--parents, teachers, state and local government and professional societies--before improvements can be available to many school children.

During the summer of 1981, regional meetings were held around the United States. Representatives of many of our projects and other experts discussed these projects and other innovative activities and tried to predict what changes could and should be expected in the next ten years, given the availability of computers. At a final retrospective gathering which included representatives from all four meetings, the participants stated that in addition to creative ideas, the following conditions were also necessary before a real improvement in mathematics education could be realized:

High quality reliable hardware, software and courseware are needed in sufficient quantities in schools if computing is to have an educational impact. Hardware that works consistently, software that is easy to use and reliable, and large quantities of high quality courseware are all needed so that the computer can be used consistently and productively by all teachers and students.

Educators--teachers, administrators, school boards--need to be informed and educated in these enhancements and changes in mathematics education and in the use of the computer itself (computer literacy).

Support structures must be created or fortified to offer assistance and continuing education for teachers in the classroom. Teachers need continuing support in the use of hardware and software (telephone hot lines or "800" numbers to experts, for example) and continuing assistance on creative and effective ways to use the computer in class.

The knowledge and potential of this approach to mathematics education should be available to all citizens. Knowing both mathematics and computing provides access to high level jobs and professions. All of our citizens should have access to these new skills.

The programs under which these projects were supported are about to--for all intents and purposes--end.

I would like to share with you some thoughts about what the coming challenges to our country now are--challenges that you must meet. I will also discuss some of the tasks that we at NSF have worked on and some that are still undone, and suggest some questions based on our experiences for you to ask yourselves as you organize to meet these new challenges.

The United States vis-a-vis Other Countries

The United States, while still the leader in computing, is facing strong challenges from other countries.

What are these other countries doing to take advantage of the opportunities promised by computing?

The Japanese--perhaps the closest behind us--have adopted an agressive national strategy and seek to dominate the high technology market sometime in the '80's. The Japanese government facilitates these developments by strong planning and financial aid such as no-interest loans.

The Japanese strength in electronics is in production; they are reluctant to invest their money in high-risk research and development. They prefer to enter the market later. However, once there, their efforts can be devastating. The industry is still anxiously awaiting the introduction of a Japanese-manufactured personal computer. And the United States worries that its preeminence in computer manufacturing may pass to the Japanese just as manufacturing of television and automobiles have in the past.

One of the reasons for these Japanese successes is that Japan is graduating many engineers--with a population only about half the size of that of the U.S., Japanese universities graduated nearly 87,000 engineers last year compared with only about 63,000 engineering bachelors degrees granted in the U.S.

The Europeans are also supporting the growth of their technology-based industries. The German government will spend $135 million dollars over the next three years to help companies apply microelectronics to new products. The French will increase R&D spending and even the British government will underwrite projects in information processing. The British are not confining their actions to the British Isles. The British are supporting the development of one of the world's most advanced microelectronics facilities--in Colorado Springs.

In all of these countries, educational activities are also being supported. The French program in national computer literacy is still going forward, though under the Socialists, now; and the British have undertaken a program to assure that microcomputing is available in every high school. The French and British will have a microcomputer in every high school this year. The BBC is working on a television program complete with a British-built home microcomputer, to teach computing via television.

The U.S. remains both the largest producer and consumer of computers--but for how long?

The Computer Revolution and Education

Education plays a key role in the transition of our economy to these new knowledge-based industries. How is our educational system meeting these challenges?

The pressures brought on our educational structures by these information and communication technologies have been great.

So far, the pressures brought by students seem to be those of rising expectations.

Students are coming to school with more information, more access to mass communication, and are generally more affluent and mobile than were their parents. But, several recent studies suggest, many of these students are finding school boring and even unnecessary. They seem to want more out of school than they are getting.

The impact on teachers and faculty has been dramatic and is primarily financial and career-oriented. Many of those with the requisite abilities and training are leaving or never entering education.

Chairs of departments of computer science go vacant, students leave school after receiving their bachelor's degrees in computer science and engineering to take high paying salaries in industry. Many students with talents in mathematics and the sciences don't enter teaching anymore. The average entry teaching salary is $12,000; the average entry salary in computing with a B.A.

Degree is $20-22,000. Many assistant professors make less money than the undergraduates they graduate. Many mathematics and science teachers leave the schools for industry. The net effect of the computer explosion has been a drain of talented people away from the educational system. The impact of technology on the structure of education, however, has been slight.

Though the computer is beginning to enter more and more classrooms, it is frequently seen as an add-on activity, not as a means for changing what goes on in the classroom or improving educational productivity. In higher education, in addition to these intellectual opportunities the computer in conjunction with other communications technologies has the potential for extending the boundaries of the campus and servicing whole new communities of adults who need access to life long learning. Yet these new structual approaches remain isolated experiments at places such as the University of Mid America and the Open University at the University of Maryland.

However, as Peter Drucker states in a recent Wall Street Journal article, "The biggest 'infrastructure' challenge for this country in the next decade is not the billions needed for railroads, highways and energy. It is the American school system, from kindergarten through the Ph.D. program, and the post-graduate education of adults."[5] He says its not that education has failed but instead has become more important for participation in the work environment.

I do not mean to suggest that the task is easy, these challenges come at very difficult times for education. With shrinking school populations, rising costs and increasing antagonism from the populace at large our educational administrators already have much to keep them busy. If we do not attend to these concerns, however, education is not only (in the words of JCR Licklider)" missing an opportunity, it is failing to discharge a responsibility."[6] It is only through education that we can develop the human resources necessary for the United States to maintain its preeminence in the communication and information technologies.

The Role of the Federal Government

How has the Federal government reacted to the international challenge? Do they see the challenge at all? And what is their view of the importance of education in meeting this challenge, science and engineering education in particular? Do they see a Federal role in science and engineering education?

For a year now you have been reading about the budget cutting, supply side economics and the general assault on Federal programs. However, in my travels about the country, I find that many people believe that "good" activities (which many believe include NSF's science and engineering education programs), will remain untouched, particularly with the emphasis on military preparedness since the military is now so technology-intensive.

That however, has not been the case. NSF's FY 81 budget was 80 million under President Carter and was scheduled to be increased to $130 million. NSF's science education budget this year (FY 82) will be 20.9 million with $15 million for fellowships and no new starts with the remaining funds. The future budget years look much more bleak. In effect innovative research,

22

development, training, equipment and communication programs have been eliminated. This is the result of a fundamental philosophical belief of this administration that the Federal Government has no role in science education. Science education has suffered more than the Department of Education and the arts and humanities which had major budget reductions.

I would like to share with you the statement of the testimony of the President's Science Advisor George Keyworth before the House Committee on Science and Technology, December 10, 1981. For some time after the budget cuts, we believed that the Administration did not understand the importance of the Science education, but Dr. Keyworth states,

> "Certainly one of the critical parts of the science and technology ingrastructure is the nation's education system." And he sees the shortages in technical fields: "We are, today, experiencing shortages of qualified personnel in several specific areas--notably computer science and some fields of engineering--and are told that these shortages may extend to additional technical fields in coming years." And the strains on universities: "Not only are the universities finding it increasingly difficult to recruit and retain faculty, but they also find it increasingly difficult to attract graduate students because industry is quite willing to hire BS or MS engineers and computer scientists at high salaries."
> But he concludes "...although this situation has serious national implications, it is primarily one of a marketplace working as it should, and does not require a massive Federal response."
>
> We thought that the Administration did not see the true condition of primary and secondary education in the United States, but Dr. Keyworth states: "Underlying current problems with the quantity and quality of trained scientific and engineering manpower is the weakness, overall, in our country's elementary and secondary school preparation in mathematics and science."

Or the centralized support education receives in other countries:

Contrast this practice with that of other industrialized countries. In Japan nearly all college-bound students take three science and four math courses during high school. West Germany is similar, and emphasis in the Soviet Union upon science and technology in elementary and secondary schools is ever greater.

While I don't necessarily endorse the philosophies that guide those other systems, I am impressed with their pragmatic commitments to science education. And it seems obvious to me that this is an area in which our system is devicient. The causes are not all clear. But, I suspect that the most important determining factor is the public attitude toward education and toward science and technology.

The conclusion of this Administration is that science education in the United States is not a Federal problem:

"Despite the Administration's real concerns in this area, improvement in this unfortunate situation is the responsibility of the schools themselves and of the communities that run them. It is up to the state and local governments to decide what kinds of specific resources should be provided for science and mathematics education in their schools--and to provide them. This diversified responsibility should also encourage the local assistance and participation by industry and professional groups in developing adequate programs in science and mathematics education." [7]

These policy statements are being translated into practice.

Hence under the New Federalism the burden falls on you--the states and local governments to prepare for these challenges as best as you can. It is now you who carry the complete burden for R&D and particularly curriculum development to help all of our citizens use computers effectively.

State and Local government support of R&D in computers in education.

It is unlikely, at least in the next several years, that there will be any new Federal investments in research and development in the use of computers in education. All of you, therefore, should be thinking about what your state and local governments could be doing to establish plans for assuming these responsibilities. I would like to suggest a series of questions for you to ask, first of all yourselves. Many of these, however, are also questions that others both inside and outside the government will ask you if and when you petition for the establishment of new or expanded R&D programs.

Why is R&D in curriculum development important? Regardless of what the Federal government or even the state governments do, I believe that computers will continue to enter our classrooms, especially the classrooms in our wealthier schools.

Teachers will spend, if necessary, their own time and money to learn about computers not just for their students, but also for themselves--computer-training (especially if you have a math or science background) means job security or (many times) a new, higher paying job.

But what do you do with a computer after the kids have learned to program? What do you do with a computer when the kids are too young to do conventional programming? Some teachers have the time and energy to do their own curricular development in computing. In fact, a recent issue of Electronics Learning stated that teachers preferred to do their own curricular development.

I can understand why teachers prefer their own work over much of what is commercially available today. Commercial publishers tell us that they provide what is basically drill and practice material because "that is what teachers say they want." However, I am impressed that many publishers do recognize innovative ideas when they are developed far enough for them to be demonstrated. Everyone, from bureaucrat to student, who has seen Sharon Dugdale's Green Globs Game, has realized that it is important and valuable--a true example of "the orders of magnitude difference" cliche.

I believe that many of these better products will appear in the next few years. If you think that you can do better than some of these commercial products you will see soon, (and I'm sure that some of you can)--start your own business doing the summer months.

It is only through extensive experimentation and the building of prototypes that we can effectively explore this media; that we can surpass the drill and practice and page turning programs that are too frequently in our schools. It is only through the process of encouraging people to experiment and try out new ideas that these breakthroughs occur.

26

What does it cost to run an R&D activity?

Let me give you some examples of costs of past activities. The mathematics prototypes that I showed you today represent a Federal investment of over four million dollars. With that we supported 36 projects. The Minnesota Educational Computing Consortium has a total annual budget in 1980-81 of $7 million; approximately 300,000 is for R&D investments. Although that includes some Minnesota $&D funds, MECC in the past relied heavily on NSF support and on private foundation support for R&D. The Apple Corporation through its Foundation for the Advancement of Computer Aided Education has invested in many projects, chiefly through the provision of Apple hardware. Atari has also just announced a $250,000 grants program.

I believe, however, that to run an effective R&D activity even on the state level would require a minimum of $500,000/year; one million would be preferable. This is of course staged with smaller costs in earlier years building up to larger investments as ideas mature. I'll be glad to tell you more about this in the post session if you like. There are techniques to use that will lower the investments required, but they have risks, too. The most important thing to remember: don't over promise for a small amount of money.

However, as many state governors are beginning to find, the responsibilities for former Federal activities have been transferred to the states faster than funds, faster than new state taxing activity, and faster that the states have developed the determination, let alone, the staff and departments, to implement these new tasks.

One response to the general decline of R&D money has been to create relationships between academia and industry. MIT has its independent Whitehead Institute for Biomedical Research staffed mostly with faculty from MIT, Stanford has a new Center for Integrated Systems supported by contributions from at least 16 industrial companies such as Hewlett Packard, Texas Instruments, IBM, General Electric and AT&T. And, Arizona State has its new Center for Excellence in Engineering though the intervention of Gov. Babbit and your state legislature. Similar activities are taking place at Rensselaer Polytechnic, Carnegie Mellon and North Carolina.

I am strongly in favor of these cooperative efforts. I believe that they build better understanding and cooperation among these different cultures in our society. I hope that we soon see a center devoted to R&D in educational applications in computing. However, dollars buy very powerful influence over their receivers. What start out as interesting avenues of research are harder to abandon when they are favorites of a corporate sponsor. The potential for conflicts of interest can increase as we have seen in the recent University of California at Davis case, in which a prominent plant geneticist's university research responsibilities and his commercial interests were seen to be too closely related.

What we may be building is a new complex in our society--the government-industrial-academic complex--and the real price for continued R&D may be our academic freedom.

Along with this cost question go a host of operational issues in conducting R&D:

- How do you identify topics with promise?
- What makes a good project?--What elements should be in a good proposal?
- Where will you get staff to operate such a program given the strong industrial demand for people with computing skills?

How do you know if what you have supported is any good?

Contrary to popular myth, computer based learning activities have probably been evaluated more than any other educational activities.

In 1970 Andrew Molnar wrote "probably no area outside the Department of Defense has received more intensive study than educational media." [8] Generally, evaluation can be divided into formative evaluation--allowing change and improvements in the product as it is being developed, summative evaluation--evaluation (frequently by a third party) after the educational product is complete; and meta-analysis which is an assessment of several evaluations.

Formative evaluation has so far been the most useful in computer-based projects. Using the computer enables the improvement of some sophisticated tasks such as algorithmic problem solving and the building of intuition about a process; skills which have always been difficult to evaluate. Evaluation, therefore has been a challenge.

However, the evidence in support of computers in instruction seems clear.

In a meta-analysis, James Kulik at the University of Michigan studied 59 independent quantitative evaluations of college level use of computing including tutoring, computer-managed teaching, simulations and programming. His findings are:

1. The use of computers makes a small but significant contribution to college teaching.

2. It has small but positive effects on the attitudes of students toward instruction and toward the subject matter.

3. Only 2/3 of the time is required to gain the same level of competence.[9]

Similar findings occur in his study of precollege computing, forthcoming.

Despite the existence of all of this information, however, you will still face demands for "proof" despite the fact that such proof is not used for new textbooks or other forms of instruction. Do some thinking about the evaluation issue.

How will you find out what is going on in other states and localities? Many new publications continue to appear about the use of the computer in education. However reading all of the publications on computers now takes a significant amount of time. And how do you tell what material would be useful for your school? Lack of protection for software leads some publishers to refuse to let schools test their materials.

As the potential for making money increases developers are also more and more reluctant to share their materials, sometimes even with each other, before it is ready for publication. Though, I believe this is in the long run the best way to make materials available to the greatest number of people, it results in new ideas being shared later. We progress at the R&D level at a somewhat reduced pace. Overall, the noise in the system is increasing rather than decreasing which is ironic for an industry that shows so much potential for increasing the amount of high-quality information available to everyone.

I don't know the answer to this question. I believe that it involves not only better communications but also legal considerations such as patents and copyrights and also the development of software and hardware that makes copying more difficult. If people felt that they had more protection they might be more willing to share insights, discoveries and even prototypes earlier.

Solving this problem, however, is critical to decreasing duplication that is already taking place. We must learn to stand on the shoulders of others who have worked before us.

How will you handle political and other outside influences?

I believe that one of the greatest successes that we have had at NSF is that decisions on projects are made on scientific quality and merit. I have never, nor do I know of any program officer who has ever been the object of political pressure.

This fairness as well as the appearance of fairness is mandatory for encouraging the best proposals and the best ideas.

One price we pay for that, however, is that in times of crisis, our friends in the political process are far fewer than those with closer political ties--and we have paid a dear price for our purity.

How you develop your relationship with the power structure will be an area of high importance.

How will you handle the growing division in our society between the computer-rich and the computer-poor?

One of the more frightening dangers from the increasing use of the computer is not only that jobs are open to those with computer experience, but also that the use of the computer itself is mind expanding; that it increases ones mental powers.

And yet in a shrinking public school system whose percentage of minority students is increasing, we must take strong steps to assure that this new tool does not increase the separation between the rich and the poor.

How, in other words, can you keep the best of our methods but eliminate the less-than-satisfactory.

How will you keep the standards of fairness and high quality and eliminate the long turnaround time and the excessive paperwork that Federal R&D has developed?

The imagination and determination with which you now respond to these new burdens and challenges will, I believe, be a key to maintaining our viability as a nation.

Conclusion

We live in exciting times. We have seen evidence of the real potential of the computer to improve our circumstances--but we have seen the dangers as well. Our present advantage is that the United States is the very best in reserach, development, design manufacture, marketing and education involving computers. Particularly in education, which I believe is the key to our remaining in first place in every other sector, we are seeing more and more evidence of approaches that will yield the long-promised "orders of magnitude difference."

Evaluation results--observational and quantitative--show that using computers in education works. Students learn about a technology that will increasingly dominate the economy and the society. Students learn better using computers, they learn faster, and they like learning better. Though there are many dangers still to be discovered, I believe there are many strengths as well. Using computers can help us to improve our educational productivity.

But Japan, Britain, France and other countries know the potential of these technologies as well, and are actively and systematically working to surpass the United States. The advantage these countries posess is that the decision-making is centralized and that they can support their goals with massive resources. This is also their disadvantage. In the United States, you as individual states and localities and you as individual teachers, who will be spending these two days improving you skills in using computers in education, have the opportunity to make significant differences in using these new technologies in your classrooms. You can take the ideas you learn here and experiment and improve in your schools next Monday. You do not have to wait years for decisions from Washington.

Continuing to explore and develop strengths and avoiding the dangers is now with the New Federalism totally your responsibility in state and local governments. It is also the responsibility of private industry.

The students who are now entering kindergarten will graduate from college in 1997. We are now educating the Nobel Prize winners, the manufacturing geniuses, the International financial czars, the world's political leaders of the twenty first century. It is your responsibility to assure that at least some of them will be Americans.

The views expressed herein are those of the author and do not necessarily reflect those of the National Science Foundation.

FOOTNOTES

[1] National Science Foundation Act of 1950, Public Law 81-507 (as amended), Sec. 3. (a).

[2] Watson, Robert F., from welcoming remarks before the NSF/NIE Regional Meeting on the Improvement of Mathematics Education Using Information Technology, Washington, D.C., July 17, 1981

[3] Priorities in School Mathematics: Executive Summary of the PRISM Project, National Council of Teachers of Mathematics, Inc., 1906 Association Drive, Reston, VA 22091, 1981, p. 1.

[4] An Agenda for Action: Recommendations for School Mathematics of the 1980s, National Council of Teachers of Mathematics, Inc. 1906 Association Drive, Reston, VA 22091, pp. 2-5.

[5] Drucker, Peter F., "As Education Heads into a 'Baby Bust,' Competition and Diversity will Prevail," reprinted from the Wall Street Journal in the Chronicle of Higher Education, May 4, 1981, p. 56.

[6] Licklider, J.C.R., "Technology in Science Education: The Next Ten Years," National Science Foundation, July 1979, U.S. Government Printing Office, GPO 038-000-00432 ($2.50), p. 3.

[7] Statement of Dr. George A. Keyworth, II, Director, Office of Science and Technology Policy, Executive Office of the President, Before the Committee on Science and Technology, U.S. House of Representatives, December 10, 1981.

[8] Molnar, Andrew R., "Media and Cost-effectiveness," Analog/Hybrid Computer Educational User Group Transactions, October, 1970.

[9] Kulik, James A. Chen-Lin C. Kulik, and Peter A. Cohen, Effectiveness of Computer-based College Teaching: A Meta-Analysis of Findings, Review of Educational Research, Winter 1980, Vol. 50, No. 4, pp. 525-544.

A portion of this talk appeared in Classroom Computer News, November/December 1981.

MicroComputers in Instructional Research

Alan M. Lesgold
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, Pennsylvania 15260

I.0 INTRODUCTION

In this presentation and the later follow-up session, I would like to discuss the importance of the microcomputer in present and future instructional research, to provide some examples to support my arguments, and to illustrate how computers are used for instructional research today. I will begin by outlining some problems that have historically been present in this field and then discuss what advantages microcomputers can bring us, either immediately or in the near future.

Before I begin, I should tell you that I have spent the bulk of my career working with large computers, which, until about five years ago, were the only type I used in research. I have a strong bias against retreating from a civilized working environment on larger systems to certain smaller ones which turn me into a low-level hacker instead of an instructional researcher. Fortunately, reasonably human-engineered microcomputer packages are starting to appear. Also, the larger machine necessary for an ideal research environment may cost $60,000 today, but at the continuing 35%/year drop in computer costs for equivalent power, even massive capabilities will cost about $1000 by the end of the decade. So, if my viewpoints seem a bit rich for your blood, rest assured that they won't be for long.

I.A The Historic Problems Of Instructional Research

Educational research, until recently, was almost impossible to do well. We did the best we could, even ingeniously at times, to produce important principles for the

---

The ideas in this paper represent approaches that have evolved at the Learning Research and Development Center over a number of years. Many people are responsible for refining these ideas, including the LRDC computer facility as well as several research projects. I thank Carol White for editing this paper, which she received only at the last moment.

design of instruction. However, those principles seemed the most successful when they systematized existing practice and least successful when they tried to explain the learning process or to prescribe new and different practices that work better. New findings from the laboratory seemed promising, but by the time they were interpreted by teachers outside our control, they seldom made a difference. While there have been a few marvelous success stories, it has, until recently, been easier to justify instructional research by pointing to the severity of schooling problems than by citing specific findings that have paid off in better instruction.

There are several reasons for this. First, the task is complex. In the physical sciences, it is possible to theorize about ideal environments and to control real environments to the point where ideal theories almost work. Similar controls are not possible in social research settings, because the required coercion would itself alter the likelihood of motivated learning. A second problem that is related to this is that the most successful theories have been complex and subtle. Since the primary mechanisms for educational experiments are instructions or training for a teacher, rigorous, replicable tests of complex theories are not readily performed. The third problem, again related to the other two, is that like the physician who takes the oath of Hippocrates, we also are bound to do no harm. We are very limited in the extent to which we can intrude into the instructional settings of greatest importance, such as the classroom. Let me elaborate each of these problems a bit, so we can see how computers can help solve them.

I.A.1 Theory Complexity -

Instructional theories must be ecological in nature. That is, they must specify the underlying principles of learning in the complex, multifaceted world in which we live. In such a world, many "entering variables" exert great influence on children's learning. We have done a pretty decent job in quantitative analysis of these entering variables, but now our theories want to speak in qualitative terms. For example, recent work done by Voss at LRDC and by others elsewhere has shown the importance of prior knowledge in understanding, problem-solving, and other activities we like to see in schools. The obvious payoff of such findings would be theories that told us exactly what the learner should encounter next in order to expand what he already knows.

But, today, we are not very good at testing children's learning efficiently or keeping detailed logs of their learning experiences. So, we make inadequate use of our knowledge of how new learning depends on the specifics of what the child already knows. The "pretesting" we can do is so inefficient that it is often the case that the more of it we do, the less learning occurs--testing time is stolen from instructional time. If we had better methods of watching the student's daily progress and activities and had better record-keeping systems, we might reach the point where we could really use theories which make detailed predictions about prior knowledge effects. Today, we can't use such theories yet--indeed we can't even test them adequately.

I.A.2 Identification Of Theory Constructs With Observables -

This brings up the second problem, the identification of both treatments and predictions for learning with observable events in the classroom. In order for a theory to be testable, its central components must closely correspond with observable aspects of the instructional setting and the student's performance. There are minimal statistical requirements for this, which have to do with the relationship between estimated parameters for a theoretical model and observables which the model can predict. However, the problem goes beyond these minimal requirements. A theory will only be acceptable if it gives useful information about the relationships between instructional treatments and instructional outcomes. In the past, because of the difficulties involved in keeping track of such relationships by hand, we have used gross summary measures of outcomes and equally gross specifications of treatments, along with warnings that the experimental instruction must be "motivating," "artistically designed", etc.

If we are to make significant progress in building instructional theories, more specificity will be needed, in both basic and applied field settings. The microcomputer is an important key to gaining this specificity, as I shall discuss in a second.

I.A.3 Avoiding Unnecessary Intrusiveness -

The third problem we face when we get down to serious instructional experimentation is a social Heisenberg principle. We cannot be in a classroom making notes on a clipboard without changing the situation. The immediate presence of researchers in a school creates a hothouse effect. Teachers sometimes try harder. Some revert to practices they think researchers will like. Some become antagonistic. Some leave. Students are similarly influenced. Some try harder, and some hang from the chandeliers.

Nonetheless, excellent classroom observational research has been done by a number of researchers. It is, however, expensive. The computer offers some possibilities for cost-cutting. Whenever instruction is offered by computer, or even partly by computer, large amounts of priceless data are continually passing through the hardware. If microcomputers could have multiple personalities that operated in parallel, then perhaps a research assistant personna could keep intelligent records of the instructional interaction. Such record-keeping could be completely unobtrusive. Further, it could also help the student by providing the basis for better tutorial instruction.

I.B The Capabilities Of Microcomputers

Having outlined some problems of instructional research, I will now discuss some of the capabilities of microcomputers that can help in overcoming these problems. I will first look at the basic capabilities of existing microcomputers and then turn to possibilities which will require more substantial microcomputers that will be affordable in a few years.

I.B.1 Basic Capabilities -

The basic capabilities of microcomputers lie in four areas: timing, symbol manipulation and text control, recordkeeping, and data analysis. All of these capabilities are expressions of a general symbol processing capability, but they have different realizations in existing systems and thus deserve separate discussion.

I.B.1.a Timing -

In principle, microcomputers can provide detailed timing of student performances and can also control time-critical instructional treatments, such as animations. However, not all microcomputer systems allow easy use of this capability. Computers are very complex collections of circuitry. Each little piece of circuitry has its own special function. Something has to impose a discipline on all these pieces. Generally, an important source of discipline inside a computer is a timing circuit that generates a regular series of pulses, or clock ticks. Each part of the system, when activated, can be relied upon to do certain things on the next several clock ticks. Thus, the timing circuit makes the whole system behave in a disciplined manner.

This timing capability is like the escapement on a watch. We need to add something that acts like the hands of a watch before we have a clock that can measure display times or response speed. For example, we can write a program that takes about a second to execute and then count the number of times that program executes between the time a display is shown to a student until the time he responds.

The problem, in essence, is to be sure that the program we write reacts quickly to the student's response. Many microcomputer systems have very complex operating systems that have large background agendas of work to do while waiting for anything. Very few of the commercial applications that drive microcomputer development are time sensitive, so there is little impetus for building good timing systems into personal computer packages. Such systems can, however, easily be added. However, the researcher will have to pay close attention to the details of such add-ons to be sure that the desired timing capability is achieved. Measuring response times of people is not something hardware technicians understand very well.*

I have spent this time telling you about clocks and response timing because speed of performance is the best measure we have of the extent to which a performance has been adequately practiced. The biggest change in instructional theory that I expect in the next few years is a move toward an integration of research on complex thinking and reasoning processes with simpler ideas about practice and mastery. We will need response measures that emphasize confident, proficient performance. Patterns of response speed will likely be the source of such measures.

I.B.1.b Display Control -

A second area in which microcomputers can be extremely useful is stimulus control. Children differ greatly in the extent to which they can follow instructions that involve deciding what aspects of a complex situation merit attention. That is why we use a variety of devices, from study carrels to simply pointing at specific places on a blackboard, to channel children's attention. The microcomputer allows such controls to be realized in a rich variety of ways: with flashing words, with black/white reversal, with different sizes of type, with color, and with sound. This capability is important when microcomputers are used for instruction, but it is also important for instructional research, since an important aspect of the testing of an instructional treatment is assuring that the treatment is in fact effectively provided to the student.

---

* One brief aside: Most computer logic clocking is not that absolutely calibrated--it usually doesn't matter. In particular, some computer clock circuits are heat sensitive, and some change with use. If accuracy to small fractions of a second are needed or if precision to less than about 5% is required, then use a crystal clock, which will cost an extra hundred dollars or so.

The specific timing of displays can also be controlled with a microcomputer. Words can be flashed for fractions of a second, although the basic screen refreshing cycle and the slow phosphors used in most television monitors limits this capability to flashes of 1/30th of a second or longer. The relative time for which different words, sentences, or pictures are exposed can be tightly controlled. Complex animations are also possible, but often harder to program than you might think. Overall, the microcomputer is an ideal way of controlling the details of instructional displays, and it allows research on a variety of formats that are otherwise too expensive to create routinely.

I.B.1.c Treatment Control In General -

Much of the effort that was previously spent conducting instructional research involved devising mechanisms for achieving control of extraneous variables. We would spend hours creating alternate forms of instructional materials and tests for each experiment. Not uncommonly, we would fail to control each variable of importance because the necessary proliferation of materials was impractical. This situation of knowing that a better controlled experiment is really necessary but being unable to do one is, I believe, a major cause of cynicism about the research process that we see in many of our brightest graduate students.

If nothing else, the computer is a great randomizer and materials generator. It can certainly allow individualized preparation of both display and print materials for each subject in an instructional study. We can no longer justify not having properly counter-balanced materials by simply stating that preparing them by hand would take too long.

However, the ability to secure very tightly controlled and individually tailored instructional treatments should lead us to a new view of instructional experimentation. Randomization and naive counterbalancing techniques, are, after all, the next worse approach to complete lack of control. What we really want is not to control for the factors that really influence learning but to understand them so that we can specifically take them into account in understanding the ecology of information in which a student finds himself. This, we are only beginning to be able to do--but it should be our goal!!

I.B.1.d Record Keeping -

I turn now to a significant function of microcomputers in research that might seem particularly old hat and pedestrian, namely record keeping. I think we underestimate the importance of good records in instructional research. Many otherwise tractable studies have foundered on problems of data reduction and record keeping. The reasons for this are very simple. Traditionally, the response data of subjects have been kept in raw paper form. Separate types of data on subjects have usually been recorded on separate pieces of paper. Thus, whenever any research questions arose involving the data, major amounts of boring, but intelligence-demanding work were required to code new data combinations and then have the coded data analyzed. Simple lack of labor resources often stands in the way of richer analyses of such data. Also, once studies were published, the storage space needed for the raw data was often unavailable, so that only the code sheets ended up being retained for more than a few years.

The true cost of the rich individualized testing of a single child using a complex experimental treatment can easily be $50, $100, or more, when experimenter time, researcher time, etc., are taken into account. We need to become as systematic and efficient in retaining our records as businesses are in retaining theirs. No sound business would let documents worth $50 each pile up in disorderly arrays today. They would use commercial database maintenance systems on computers to control the whole data acquisition and storage process. We need to do the same thing.

The use of database software for research is quite common on larger computers, and our colleagues in large-scale evaluation work now have direct interfaces between commercial database systems and the major statistical packages. Instructional experimenters need to find ways of keeping their data in machine readable form through all steps of the research process, from initial instructional experiments to final analyses and data storage. For smaller laboratories, this can be accomplished increasingly through the use of a single microcomputer. Database maintenance systems are now available for the most common microcomputers and for standard operating systems, such as C/PM and Unix.* I encourage you to learn to use such resources and to press for better ones. Good educational experimentation is complex, difficult, and expensive. Also, we should not be intruding unnecessarily in the classroom. For these reasons, we have an obligation to get the most out of our data. To do that, the data must be accessible, and it must not get lost. Database systems can make this possible.

---

* C/PM and Unix are trademarks of Digital Research, Inc. and Bell Laboratories, respectively.

I.B.1.e Analysis -

The final area of microcomputer use in research that I wish to summarize is data analysis. Here again, we need to concentrate on the capabilities of the computer as a data crunching slave. Data analysis is very much a practical activity. We try to get approximate answers to the questions that we have, using statistics that are practical to compute. Unfortunately, in many research ventures, especially thesis research, the rationale for doing specific analyses was that those were the only ones available in the few canned statistics packages the researcher knew about.

Think about what has happened. Before computers, statistical analyses were restricted to what could be done by hand on mechanical calculators. Then, when computers were introduced, we were the beneficiaries of a tool for doing such computations without human effort and at superhuman speed. We should then have become much more sophisticated in the analysis methods we used, since more was possible. Some of us did, but many of us took a step backwards. We stopped doing analyses by hand, since that was always error-ridden and had now also become socially demeaning. We did not, however, master the new tools. Instead, we became dependent upon the marketplace and used whatever analysis resources happened to be programmed for the computer we could access.

I fear that this is happening again. As microcomputers proliferate in research, we see advantages of keeping the data in one place throughout the research process, namely on the laboratory microcomputer. Sometimes this is appropriate, but in other instances we should be establishing file transfer facilities which allow us to use statistical resources on larger machines. As microcomputers become more powerful, this will be less of a problem, but in any case, having new resources should never be an excuse for stepping backwards into less appropriate or less reliable procedures just because no one else has yet done the programming work that is needed. One important reason for buying more computerpower than appears to be needed in a microcomputer is to increase the likelihood that good analysis tools will be available and/or that civilized environments for programming them will exist.

I.B.2 Coming Soon -- Intelligence -

Having talked about basic microcomputer capabilities for research, I now turn to capabilities that will only become possible over the next several years, namely intelligent research environments. At first blush, the possibility of even more complex computers in educational research might seem threatening. However, the likely result will be systems that are much easier to use and much more useful. This is because of a confluence of three factors, namely:

- computers will keep getting more powerful, while the price of equivalent computer power will drop by perhaps 35% per year.
- major innovations in such fields as artificial intelligence, database systems, and computer-assisted design have led to ever richer arrays of tools for supporting the development of very large and complex programs using simple languages. These systems include many tools for preventing bugs and simplifying program development and maintenance.
- the office automation revolution made possible by the first two factors will provide an economic base for such systems, thereby forcing some software prices down more rapidly.

Thus, we can expect that programming will not be as painful as before. It will be easier to develop rich libraries of tested procedures from which new tools can easily be assembled.

What will this buy us?

I.B.2.a CBI With A Built-in Research Assistant -

First, it will be possible to build research assistants into instructional systems. Even today, any instructional system ought to have some capabilities available for logging the specifics of the interaction with a student. However, even a short transcript of serious conversation can be a monumental task to analyze. For this reason, there should be intelligent computer resources for analyzing a student's interaction with a computer and recording the results of that interaction.

In order for such analytic capabilities to become widespread, they will need to be portable and somewhat independent of the specific instructional modules with which they co-exist. That is, the research assistant part of an experimental instructional system will need to be a separate personality that can watch the student interact with the rest of the system.

The tools needed for supporting multiple, co-existing, semi-independent system entities are not present in the Apple II generation of microcomputers. This is because such systems are inherently single-user systems. What is needed is the ability to have several components that are independent in terms of their interactions with the operating system but can pass data to one another. Such capabilities exist in the large and expensive personal machines that are designed for use by artificial intelligence researchers--soon the rest of us will be able to afford them, too. Useful research assistant modules will also require powerful symbol processing capabilities, such as the more elaborated versions of lisp, and that is also coming soon. For now, the point I want to make is that we should be setting the goal of building both teaching and research power into instructional computer systems.

This does not mean that I propose eliminating any teachers or researchers of the human variety. Rather, I feel that the tasks that face us if we are to do instructional research of enduring practical importance are so massive that we will need all the human and machine ingenuity we can marshall.

I.B.2.b Simulating Complex Theoretical Ideas -

A very different role for computers in instruction is for simulating the predictions of complex instructional theories. Today, we espouse a principle and then test it in one of two ways. Either, we contrive a very artificial laboratory test and then blithely tell teachers that the principle apparently confirmed in the laboratory will also work in the classroom, leaving implementation details as an exercise for them. Or, we rehearse the principle to ourselves as a monk might repeat a mantra and then, after achieving enlightenment, design and test a piece of curriculum which is only loosely coupled to the principle. The alternative is to have intelligent systems explicitly and replicably use new principles to tailor instruction to specific children and then see if the prescribed instruction actually works. This would give us stronger tests of our hypotheses, a clearer means of implementing them as instructional prescriptions, and, in the trace of the computer's reasoning, examples of how the principles should be applied.

Is this a utopian vision? I don't think so. The necessary computer power will be affordable in only a few years. Now is the time for at least some researchers to be developing new approaches to prescriptive instructional theory that take advantage of new computer hardware, new software, and new principles for complex system design.

I.B.2.c Interactive Experimentation -

We are moving toward a system in which instructional principles drive the specifics of interactions between tutors (machine or human) and students. Envisioning such possibilities, Glaser some time ago called for a methodology of interactive experimentation in which principles for adapting to individual differences in real time were tested instead of static instructional treatments. Additional work is required before we can do such experiments, but the possibilities are achievable and important. First, it will be necessary to develop and test a variety of resources to support student-machine interaction. This presents a different but equally important research agenda. For example, we need to know about how the different means of interaction with instructional microcomputer systems affect learning. We need to determine which children, with which learning goals, should be asked to communicate with a computer by typing. We need to know whether a touch screen, in certain situations or with certain students, offers important advantages over game paddles or other spatial input devices. We need to know when voice output is necessary and which forms of voice output are adequately comprehensible. Many educational researchers will be needed to settle these matters.

A second research agenda involves something closer to computer science. Here, we will need people to do research on the sorts of authoring environments needed by instructional designers and researchers who want to test their ideas by embedding them in instructional systems. There are many facets to this problem, which I have discussed elsewhere,* but it boils down to having people of great talent looking at what is going on in artificial intelligence and other complex system design areas and applying the discoveries in those fields to the design of an authoring environment for instruction.

The things I've just said may sound grandiose to some of you. They aren't. Everything I've talked about is underway somewhere, but more work is needed and more viewpoints of people who know about classrooms are sorely needed. Nonetheless, I'd like to turn now to a brief examination of several research uses of microcomputers that are well underway, perhaps close to completion.

---

* Lesgold, A.M. Authoring environments for computer-based instruction. Paper commissioned by the National Institute of Education. October, 1981. Available from author.

II.0 EXAMPLES FROM READING RESEARCH

It is difficult to use small microcomputers to teach reading skills. Very young children cannot type very well. The displays on cheap microcomputers are of low quality. Having to interact with a somewhat clumsy small system may require so much of the child's processing capabilities that there is not enough left for substantial learning to occur. Nonetheless, exemplary work has been done with very small systems. I would like to briefly explore just a bit of this work.

II.A Frederiksen And Weaver's Work

The most ambitious program so far has been that of John Frederiksen at Bolt, Beranek and Newman. Working initially with Weaver, Frederiksen developed both a battery of tests of component processes of reading and a variety of training tasks, all of which were realized on a very small microcomputer. The results of this work are only beginning to appear, and Frederiksen deserves to present them himself. For our purposes, the important thing is that by developing a timing capability in his software, Frederiksen was able to use a microcomputer that costs only a few hundred dollars, with only a cassette recorder, to administer a wide variety of response time tasks and to provide several types of training in efficient word recognition and even low-order comprehension skills.

II.B Reading Games

Much instructional research centers on testing of curricular innovations. Many recent innovations, including some of Frederiksen's, are in the form of computer games. One type of reading game, with which my colleague Isabel Beck has been experimenting, is a Beat the Clock format. Children are asked to find specific words or subword components in a displayed array. A manipulable time limit can be imposed, a clock can tick, and other inducements can be provided to emphasize speeded performance. The format can be extended to include a wide variety of exercises involving the specifics of meaning and/or structure of words, subword groupings, sentences, and even larger units.

II.C Text Coherence Studies

At a more basic research level, a computer can be used to time how long it takes subjects to read individual sentences or other segments of text. A few years ago, for example, Steve Roth, Mary Beth Curtis and I* studied the patterns of reading times for sentences in discourse as a function of whether the concepts to which they referred were present in the subject's short-term memory. Different models of the time course of discourse comprehension led to different predictions about what concepts should have been in a subject's mind at the time he encountered specific sentences. Using a computer to display a sentence and then measure the time until the subject asked for the next sentence, we were able to verify that sentences do take longer to understand when they refer only to concepts that are not currently in mind. Further, in that work and subsequent dissertation work done by Roth, a partial model of the discourse understanding process has been assembled.

The uses of the computer were very mundane in this work. However, it is important to note that the work would have been expensive if we had needed to buy random access slide projectors, make counterbalanced series of slides, and use electronic shutters. With the computer, it was much more straightforward, and new studies could be quickly generated from those just finished. I want to turn now to a final possibility in reading research that we are only starting to design.

II.D The Reading Machine

By now, a number of people have given thought to the sorts of aids that microcomputers might provide to the reading process. For example, if text is presented on a microcomputer with adequate internal and disk memory, an online dictionary could be offered quite easily. Given voice output, it would be possible to have a hard word pronounced by the machine. If there were a way the reader could point to a hard word, this pronunciation and meaning service would be particularly valuable.

Of course, we can look over the shoulders of children using these capabilities by building a semi-intelligent research assistant into our great American reading machine. Suppose that small amounts of text are displayed at any given instant, with the student pushing a button when more was needed. We could then have quite a range of useful data from anyone using the machine to read a few pages of text, including:

---

* Lesgold, A.M., Roth, S.F., & Curtis, M.E. Foregrounding effects in discourse comprehension. Journal of Verbal Learning and Verbal Behavior, 1979, 18, 291-308.

- reading times
- meaning problems
- decoding problems

From all of this data, it should be possible to get answers to all sorts of questions about the specifics of a child's word processing skills.

This general approach suggests two roles for instructional researchers. Some of us will want to design classroom artifacts that are both useful to the student and a source of data for the experimenter. Others will want to take advantage of these tools as they emerge, using them for a variety of research tasks.

III.0 TRANSACTION PROCESSING AND DATABASE SOFTWARE

I would now like to elaborate on a second area I addressed earlier, namely the use of transaction processing and database management capabilities. A database management system provides an easy-to-use set of tools for storing and retrieving data and for protecting data from accidental damage. These systems also often include a rich set of tools for generating summary reports from the data they are managing. Transaction processing resources are those that allow fast, idiosyncratic updating of the database as new data are collected. Large computers have a variety of commercial systems available for database support, but microcomputers are starting to afford these capabilities, too.

Why are they important. If you buy my argument that we should be moving from an instructional research model in which small, fixed laboratory experiments are replaced by more complex, and possibly longer ranging studies, then the need is clear. It is one thing to do an experiment, write all the scores on data sheets, and analyze them. It is something altogether different to keep track of collections of data that do not come in all at once or that do not organize themselves into one neat matrix of numbers. Let me provide an example.

Lauren Resnick, Isabel Beck, and I decided in 1976 to conduct a longitudinal study of reading. We wanted to administer a variety of laboratory tasks to children as they progressed through the primary grades reading curriculum. Children were tested at intervals determined by their curricular progress. Thus, the process was event driven. Whenever a teacher reported that a student had completed part

50

of the curriculum, his name was added to the the roster of children waiting to be tested. Given the need to prompt teachers to keep us up to date on children's progress and the need to reserve testing time, just collecting the data required a database system that could automatically warn us of children who had been waiting a week or so for testing and of those about whom we had had no recent information from the teachers. The resultant data, which dribbled in over several years and a succession of assistants, had to be checked for scoring validity and consistency and then added to the database. In addition, we had to be sure that the inevitable correcting and updating of files did not accidentally result in loss of data.

When we began the study, a senior colleague at another university took me aside and told me how hard it would be. He warned me of the many long-term studies that had failed because of the problem of preserving a consistent, reliable data gathering capability. He was absolutely right to be concerned. However, by using the same database system that the university uses for payroll, we were able to make the data collection process quite manageable. A very small staff collected continual data on almost 300 children over a five year period--hundreds of thousands of data points.

On a smaller scale, the tricks of the database management trade can be carried over to microcomputers, and this is starting to happen. Without such capabilities, research with a microcomputer will be very perilous. Paper looks fragile, but, compared to microcomputer storage media, it is quite robust. Data written on a single sheet of paper is pretty safe compared to data on a diskette or cassette, either of which may be rendered useless by folding, insertion into a disk drive at the wrong time, magnetic interference, or other not uncommon problems. For example, I am told that trips on subways have been known to destroy magnetic recording media.

To summarize, when planning an experiment involving microcomputers, think hard about data preservation. Use the tools that are starting to appear. Database software will cost money, but so do file cabinets. Multiple copies of diskettes will also cost, but so does xeroxing. Be careful.

IV.0 IMPLICATIONS FOR CLASSROOM OBSERVATION WORK

A very different set of microcomputer uses arises in the area of classroom observation research. Traditionally, classroom observation studies have been done by standing in classrooms with a clipboard and checking off various categories of behavior for each child according to a complex time schedule. Afterwards, all the data had to be coded and then typed into a computer for analysis. More sophisticated possibilities also existed: one could videotape the classroom and analyze it later; also, one could use a multichannel event recorder to record the classroom events as blips on one of several timelines. The basic problem was that a reliable record of timing could not be kept by hand at the same time that complex analyses of student-teacher interactions were taking place.

This problem can be greatly ameliorated by the use of a specialized class of microcomputers, namely the data loggers that are used for such purposes as inventory control. Some of these devices can be equipped with clocks, so they can log the exact time of each keystroke. Further, they are programmable, so that they can ask in turn for the type of behavior as well as the classroom, the teacher, and the student being observed. Such programs can permit immediate error checking; the observer can be prompted for a new entry if the first one is unacceptable. Finally, more complex scoring schemes may become feasible if they can be presented as hierarchical menus of choices, so that at any instant, the observer is selecting one of only a small number of alternatives.

V.0 INTELLIGENT MACHINES

I would like to conclude by briefly discussing two applications that cannot today be realized on small microcomputers. Such applications will be affordable in only a year or two, and it seems appropriate to give you some hints of what is soon to come. Both applications involve the use of powerful personal computers that would cost about $60,000 today, but the prices are tumbling. Both also use some of the tools developed by artificial intelligence researchers in their work on intelligent computer systems.*

V.A DEBUGGY

The first example is a program developed by Brown and Burton called DEBUGGY. What Brown and Burton did was to examine the mental processes that people go through when doing subtraction. They were able to build a very complete cognitive model of that activity, and then to verify that model with a computer system. That system could take a child's specific answers to a large number of subtraction problems and then account for his or her error pattern by citing one or a few small steps of the subtraction process that were defective. In many cases, children who seemed to their teachers to be making random, careless errors could be shown to have a specific deficit.

---

* For a good introduction to artificial intelligence, see the Handbook of Artificial Intelligence, edited by Avron Barr and Edward Feigenbaum. Volume I is out and Volumes II and III are on the way. William Kaufmann is the publisher.

Almost all children's error patterns could be classified in this manner, as being due to the lack of specific steps of the process. The computer program seemed to be providing a solid confirmation of the theory. However, later work revealed that any given child might be showing one specific process weakness or bug one day and a different one the next. That is, components of the overall process seemed to come and go, to some extent. More recent work by Brown and Van Lehn has used the same simulation modeling capability to try to build a deeper, more complete model that can account for these shifts. None of this work would be possible without the availability of an extremely powerful, easy-to-use computing environment for research. There were too many possibilities for missing skills, so even the initial analysis needed a computer. Once the bugs were known to be variable over time in the same child, even richer computational theories were required.

V.B SOPHIE

A second example, also from work by Brown and Burton, involves the development of tutorial systems for teaching electronics troubleshooting skills. A program called SOPHIE was developed that contained several important components:

- An electronic circuit simulator
- A problem presenter that presented circuit diagnosis problems to students
- An interactive capability that allowed the student to ask questions about electrical measurements in the imaginary faulty circuit and to receive answers from the circuit simulator
- And, most importantly, a tutoring system.

In fact, the system was quite usable as a laboratory for training technicians. The important point for our purposes was that the tutoring system was written so that specific tutoring principles could be added or removed easily. This permitted easy research on the principles of tutoring in problem solving, which is still underway.

To close, I want to emphasize again the multiple roles computers can play in research, as controlled teaching resources, as data gatherers, as stimulus preparers, and finally as intelligent aids to the development and empirical exploration of instructional theory. It's worth spending a bit of your time keeping up with this technology. Some tools are ready for use; others will be by the time your next research project is underway.

WHY AND WHERE SHOULD MICROCOMPUTERS BE PLACED IN SCHOOLS
--PERSPECTIVES, RECOMMENDATIONS, AND RESOURCES--

Dr. Mitchell E. Batoff
Director, Center for the Advancement of Teaching and Learning
Jersey City State College
2039 Kennedy Boulevard
Jersey City, New Jersey 07305

* * Outline of Presentation * *

A. INTRODUCTION: Excerpt from an SRA publication and display of several other items

B. WHAT IS ALL THE FUSS ABOUT?

Five Choice References:

1. Running Wild: The Next Industrial Revolution by Adam Osborne. Osborne/McGraw-Hill., 630 Bancroft Way, Berkeley, California 94710, 1979. 181 pp. illus.
2. The Micro Millennium by Christopher Evans. The Viking Press, 625 Madison Avenue, New York, N.Y. 10022, 1980. 255 pp. $10.95. This book is also available in paperback.
3. The Third Wave by Alvin Toffler. Bantam Books, Inc., 666 Fifth Avenue, New York, N.Y. 10103, 1981. 537 pp. $3.95—See Chapters 14, 15, and 16 in particular.
4. Media: The Second God by Tony Schwartz. Random House, New York, N.Y. 10022, 1981. 206 pp. $13.50. —See pp. 187-197 in particular: "Communication in the year 2000." Read "between the lines" as you study the developments forecast on these pages. Realize that computers are an integral part of all or most of these services and techniques.
5. 2081: A Hopeful View of the Human Future by Gerard K. O'Neill. Simon and Schuster, Rockefeller Center, 1230 Avenue of the Americas, New York, New York 10020, 1981. 286 pp. illus. $13.95—

(O'Neill is Professor of Physics at Princeton University, originator of the space-colony concept, President of the Space Studies Institute, and author of the award-winning book about space colonies, The High Frontier published in 1977).

In the Preface, O'Neill refers to "The Drivers of Change"...five developments that he believes "will determine, alone and in combination, the course of the next hundred years."

The Drivers of Change:

1. COMPUTERS
2. AUTOMATION
3. SPACE COLONIES
4. ENERGY
5. COMMUNICATIONS

"These five are the forces that will drive the changes of the next century. The 'captains and the kings' will come and go, but the five will endure and will shape the world, unless we are destined for the final catastrophe in the brief moment of time that lies just ahead. If we survive these dangerous decades, the human race will be unkillable, because it will have begun to spread throughout the solar system, our pathway toward the stars. Computers, automation, space colonies, energy, and communications: the five. And unless we do something violently stupid with our future, the eternals of hope and love and laughter will still be there too, and will accommodate all the might of the five to everyday human affairs just as successfully as they have already tamed the automobile and the jet airplane and, even, the telephone."

C. WHAT ARE THE IMPLICATIONS FOR EDUCATION?

1. What is in store for the short-term future, 1982-1984? How about the next decade?
    - a. Display and comment on Infoworld
    - b. Comment on 60-70 other microcomputer periodicals worldwide
2. What are the implications for education of recent and future developments in the new information technologies (videodisc systems, videotext systems, word processing systems, cellular communication technology, electronic blackboards, teleconferencing systems, and MICROPROCESSOR-MICROCOMPUTER TECHNOLOGY)?
    - a. Curricular changes/additions
    - b. Facility-equipment changes/acquisitions
    - c. Administrative changes
    - d. Continuous inservice professional staff development
    - e. Modifications in preservice teacher preparation

D. WHAT IS BEING THRUST UPON TEACHERS AND ADMINISTRATORS

1. "Just another audiovisual aid no different than an overhead projector." ??? XXX
2. "A panacea for teaching and learning." ??? XXX
    - "grade one counting programs" ???
    - "software programs to teach or reteach parts of speech such as nouns, verbs, adjectives and adverbs ???
    - "The microcomputer has proven to be a valuable tool in helping to provide initial and ongoing assessment in an early identification program for students in junior and senior kindergarten. Colour, shape, quantity and quantity-numeral discrimination are evaluated in these easy to operate programmes. These young students take readily to the computer medium and seem to enjoy the visual and audio-visual feedback." ???
    - "Drill and practice materials in mathematics for grades 3-6 have been used in Alberta Schools for the past three years." ???
    - "During this session we will examine four programs that can be and have been used with success in the Primary Grades. The programs are:

        INTRO which introduces letters and numbers to children
        LETTER REC which drills recognition skills
        SYMBOL which drills the classification and discrimination skills
        MATH DRILL which drills simple math." ???

E. WHAT IS IN STORE FOR TEACHERS AND ADMINISTRATORS?

1. A chaotic mess with at least three broad trends discernible in the near future:
    - a. D_______then, publishers and hucksters now.
    - b. Sensible and valuable applications of microcomputers.
    - c. Expansion of the capabilities of microcomputers into the domain of minicomputers.
2. An exciting learning environment for all concerned if wise policy decisions are made by people with influence.

F. WHAT IS AT LEAST ONE PREREQUISITE FOR THE MAKING OF WISE POLICY DECISIONS?

A well-thought-out overview—or map of the terrain—to help place microcomputers in perspective with regard to educational uses, purchase decisions, personnel, and the like.

G. USES or APPLICATIONS OF MICROCOMPUTERS IN EDUCATION

1. Administration 2. Instruction 3. Research

H. ADMINISTRATIVE APPLICATIONS OF COMPUTERS IN EDUCATION

1. School attendance
2. Career and guidance information and matching
3. Library circulation, records, and reports
4. Test scoring
5. Grade reporting and distributions
6. Instructional management
7. Curriculum development via "electronic cutting and pasting"
8. Electronic mail and memos
9. Record management via electronic filing
10. Property management via electronic inventory control
11. Management of mailing lists and labels
12. Management of petty cash
13. School and district financial management
14. Scheduling/Timetabling
15. Text editing/word processing of letters and documents
16. Automobile parking permits
17. Compilation and analysis of athletic team statistics
18. Compilation and analysis of basic competency records
19. Management of business/personal records
20. Compilation of honor roll and rank in class tabulations
21. Management of school calendar
22. Scheduling of committee meetings
23. Management of student activity records
24. Management of student deficiency reports
25. Management of an equipment inventory
26. Management of textbook inventories
27. Management of student process reports and permanent records

I. SOME IMPORTANT CONSIDERATIONS REGARDING THE MANIFOLD ADMINISTRATIVE APPLICATIONS

1. Proper programming
2. Supplemental storage capacity
3. Adequate printer for the job at hand
4. Will the hardware you have selected withstand the heavy usage it is being subjected to round the clock every day of the year?
5. Subtle but significant contributions of administrative applications to computer awareness and computer literacy of total school community

J. ADVANTAGES AND LIMITATIONS OF MICROCOMPUTERS COMPARED TO MINICOMPUTERS AND MAINFRAME COMPUTERS

1. Initial cost
2. Operating costs over a period of time
3. Personal control and convenience
4. Breakdowns, down-time and remedies
5. Ready access to a large number of programs in external memory
6. Ready access to a large data bank in external memory
7. Other considerations

K. INSTRUCTIONAL USES OF MICROCOMPUTERS IN EDUCATION

1. Microcomputer Games at the Primary Level and Beyond
2. Computer Assisted Instruction (CAI)
    a. Tutorial
    b. Drill and practice

- What does the research on CAI show? As of 1981, is it effective?... effective for what?...at what level?...for what type of pupil?
- Is it cost-effective? Are there less expensive ways to accomplish designated objectives (such as drill and practice in arithmetic)? - Consider devices such as System 80, Classmate 88, Speak and Spell (and other similar devices), and flash cards!
- Is a particular form of CAI and its subject matter linkage pedagogically sound? Is the use of CAI for a particular purpose a step forward or a step backward? E.g., see Papert on the learning of mathematics.
- Consider the multitude of problems associated with CAI software development; and the location and evaluation of quality software by educators.
- How and where does the CAI software fit into your curriculum?

* * * * Move cautiously in the area of CAI for at least four reasons:

a.

b.

c.

d.

* * * * Some situations that justify the use of microcomputers for tutorial or drill and practice (i.e., CAI):

a.
b.
c.
d.
e.
f.

3. Computer Managed Instruction (CMI) in the Classroom

a. to assist with test construction and scoring
b. to help individualize learning
c.
d.
e.

4. The Microcomputer as a Problem Solving Tool in a Wide Variety of Disciplines

   a. To retrieve and use information from huge data banks (E.g., library or media center application, or home!)—Personal tool for accessing information.

   b. To solve complex or tedious mathematical problems.

   c. To help present mathematical graphs to pupils (E.g., to demonstrate the solution of simultaneous equations graphically, or, as a student oriented tool, to verify or investigate graphs in a calculus class).

   d. To help solve complex problems in the natural and social sciences via simulation (E.g., problems associated with population, pollution, radioactivity, business and financial matters, and the like).

   e. To help demonstrate subject matter—via simulation—that is difficult or impractical to demonstrate in the laboratory or field (E.g., a simulation of Millikan's original oil drop experiment).

   f. To help present physical science concepts graphically (E.g., a plot of electric field potential around a charge).

   g. To collect and analyze data in the physical science laboratory via microcomputer interfaced with various laboratory apparatus (E.g., a solar panel).

   h. To control devices and conditions via microcomputer interfaced with apparatus/equipment (E.g., control of water temperature).

   i. To help solve problems associated with written communication—via a word processor; to facilitate writing.

   j. To compose music via a microcomputer and music synthesizer.

   k. To solve problems associated with art, graphics, and animation.

   l. To control typesetting in a graphic arts department.

   m. To extend the range of usefulness of an interactive videodisc player (E.g., as found in Discovision).

   n. To help prepare Business Education students for the rapidly approaching office of the future (e.g., electronic mail).

   o. To help prepare Industrial Arts/Vocational Education students for digital electronic applications that are used in industry today (E.g., computer control of electric motors, selective computer control of a security system, and computer control of telephone dialers).

—The above fifteen examples are problem solving applications in a variety of fields. It is important for students to have experience with microcomputers in physics, chemistry, biology, mathematics (including statistics), economics, demography, language arts, journalism, music, business practice, art, graphic arts, and industrial arts shop because the computer is rapidly becoming an integral part of each of these disciplines or fields of study. Several of the examples (a-o) illustrate the use of the microcomputer as a teaching device. However, in either case we have an entirely different use of microcomputers than in CAI or CAL.

The microcomputer is a tool that will be as commonplace as a slide rule was two decades ago and as pocket calculators and pencils are today. It will pervade all of the sciences, engineering, the health professions, such as nursing, medicine, and dentistry; law, business, the arts, and more.

5. The Microcomputer As An Object of Instruction to Develop Computer Awareness and Computer Literacy and Career Potential

   a. Nature of computers/computer system components

   b. Programming in one or more languages leading to a functional skill at computing ⟶ problem solving and applications in various fields

   c. Identification, use, and evaluation of commercially available programs

   d. Microelectronics of computers

   e. Mathematics associated with computers

   f. Applications and limitations of computers

   g. Societal implications and concerns; impact upon society

   h. The multitude of career opportunities associated with computers and the different preparation required for different types of jobs. See the 30 pages of advertisements for positions in the Business Section of the New York Times (E.g., Sunday 10 January 1982 or any other Sunday).

   ---Luehrmann's calculations with respect to computing experience in school and income advantage.

   ---Computer Awareness and Computer Literacy materials available now or in the near future:

   - For a comprehensive directory of resources see the Appendix to this outline.

6. The Microcomputer As a Device to Enhance Intellectual Development of Children

   See— Mindstorms: Children, Computers, and Powerful Ideas by Seymour Papert. Basic Books, Inc., 10 East 53rd Street, New York, N.Y. 10022, 1980. 230 pp. $12.95

L. THE MICROCOMPUTER AS A RESEARCH TOOL

1. For search/retrieval of information
2. For learning about pupil learning (E.g., Carl Berger's research at the University of Michigan; Papert's work with children and LOGO)

M. WHAT IS NEEDED IN THE DECADE AHEAD AND WHY: SOME SPECIFIC RECOMMENDATIONS

1. The development of K-12 Computer Education programs (i.e., curricula) in the schools starting with a cadre of interested, enthusiastic teachers at several levels. These teachers need not necessarily be in the mathematics department although they might be.

   The situation around the country, with respect to microcomputers in schools, is somewhat chaotic and amorphous, especially in elementary and middle schools. In many places there appears to be little direction or well-thought-out purpose as to why the microcomputers are in a particular grade or classroom and where their use fits into a total curriculum. Is the taxpayer's money being spent wisely?

   In my opinion, what is needed over the next 2-3 years is the development of K-12 Computer Education programs. The Computer Literacy Course at the 7th or 8th grade level (mentioned on this page, above) would fit into this K-12 Curriculum as one important component. But other things should happen at other levels. Moreover, it should all fit together in some meaningful sequence.

60

One example of an effort to build a Computer Education Curriculum, K-8, can be found in the Cupertino Union School District, Cupertino, California. This endeavor is reported in the September 1981 issue of The Computing Teacher, Vol. 9, No. 1, pp. 27-34. The eight pages on the Cupertino Computer Literacy Curriculum are reproduced here with permission. (See Appendix).

IMPORTANT RESOURCE

Anyone embarking upon the development of a K-12 Computer Education Curriculum should certainly consult the following significant document:

Course Goals for K-12 Computer Education to help answer the question: "What Should Students Learn in Computer Education?" A resource for selecting and communicating student learning outcomes as a basis for educational planning and evaluation. Developed by the Tri County Goal Development Project, a consortium of more than 55 school districts in the Portland, Oregon, Metropolitan area. 1979. 214 pp. soft cover. To order, send a check or purchase order for $14.40 to: Commercial-Educational Distributing Services, Post Office Box 4791, Portland, Oregon 97208. (Price effective 1 July 1981 and subject to change). The Project was promoted by the Northwest Regional Laboratory, 710 S.W. Second Avenue, Portland, Oregon 97204. Funding came from a variety of sources, including the U.S. Office of Education, Department of HE&W.

Here are five important places to begin concentrated efforts while a K-12 Computer Education Program is being developed. Each of these five endeavors will fit into the K-12 Computer Education Program as a significant component of the total plan:

(a) Incorporate microcomputers and microelectronic projects into Gifted and Talented Programs at the K-6 or K-8 level.

(b) Establish a Computer Literacy Course at the grade 7-9 level with the development of computing as a functional skill as a primary objective/outcome of the course. This should be a laboratory course with a lecture-discussion component, and required of all students. An indispensable part of this course is a computer laboratory providing one computer for every two students. Arthur Luehrmann's calculations on the cost of a laboratory are as follows: "Taking an average figure of $30,000 and spreading it over a five year amortization period, the annual equipment cost comes to $6000. One teacher, teaching five sections of a one-semester computer course, can teach 150 students per semester, or 300 per year. Thus, the added cost per student for this one laboratory course is about $20." By September 1982 there will be at least two quality resources available for teaching such a course. See References #10 and #8 in the Appendix.

(c) Microcomputers should be incorporated into all the disciplines—as a tool for problem solving—from grade 7 ⟶ 12. See Section 4, on page 5 under K. If the Computer Literacy Course (e.g., A First Course in Computing) were taught in seventh grade, then students would be well equipped to utilize these programming and problem solving skills in all their courses from grade 7 until graduation (plus a great many out-of-school applications).

(d) Establish advanced electives at the high school level (e.g., advanced BASIC, Pascal, Graphics and Animation, Robotics, Word Processing for Non-business Students, more).

(e) Support experimental work with LOGO and Papert's ideas in the primary grades.

---Other lower priority initiatives that fit into the emerging K-12 Computer Education Program:

(a) (b) (c)

2. The Development of Computer Awareness and Literacy programs (curricula) in colleges and universities starting with a cadre of interested, enthusiastic teachers--whereever they may be in the institution. The effort need not necessarily begin or end in the Computer Science Department (although it might). The members of these departments often have interests, experience, and expertise outside of the mission under consideration here. Five important places to begin concentrated efforts:

   (a) The establishment of a Computing Laboratory under the aegis of a person or group in the Faculty of Arts and Sciences.

   (b) The establishment of a Computing Laboratory under the aegis of a person or group in the Faculty of Education and Professional Studies.

   ----Each of the foregoing will probably serve different target audiences and each can be easily justified in terms of its particular mission. Both should be patterned after the following general model: 15 microcomputers interconnected with a Corvus-type hard disk external memory, constellation unit, and peripherals (disk drives, printers, plotters).

   (c) A required Computer Literacy course including a laboratory component. This course should be taught by a master teacher wherever he/she may be found.

   (d) Microcomputers should begin to infiltrate every department in the college or university and should not be restricted to the two computing laboratories mentioned under (a) and (b). One might follow the early model of Dartmouth and Waterloo but translate the terminal set-up to micros (or terminals plus MICROWATs). This effort should begin with faculty who have expressed an interest in microcomputers and would like to incorporate problem solving applications into their courses.

   (e) High quality professional development seminars should be offered to faculty and staff--on an ongoing basis throughout the year.

3. Other initiatives

   (a) Awareness workshops for teachers and administrators.

   (b) Intensive Professional Development Seminars and short courses for teachers and administrators.

   (c) Establishment of Microcomputer Resource Centers for use by the total community: teachers, administrators, pupils, parents; college and university faculty, staff, and students. See "N" on page 9.

N. MICROCOMPUTERS: AN INTRODUCTION AND GUIDE TO THE LITERATURE. INFORMATION SOURCES FOR THE ESTABLISHMENT OF A BASIC REFERENCE SHELF OR MICROCOMPUTER RESOURCE CENTER. HOW TO SET UP A RESOURCE CENTER.

Overview of Presentation

Some remarks on microcomputer hardware and recent developments of great importance. Books for teachers relevant to microcomputers, personal computers, or desk-top computers (some choice selections). Some good books for children and adolescents. Publishers/distributors of books on microcomputers, computing, and related topics. A guide to fifty periodicals devoted to computers in general and microcomputers in particular. A baker's dozen of particularly useful and important periodicals. Canadian, British and Australian sources. References to some interesting and significant articles in non-computer periodicals. Key organizations that are important resources. Selected non-print media. User publications and user groups. Computer stores. Computer clubs. Some key human resources around the USA and Canada. Software sources and clearinghouses for the identification and evaluation of quality software.

For information on the above presentation, contact Mitch Batoff at CATALYST, Jersey City State College, 2039 Kennedy Boulevard, Jersey City, New Jersey, 07305 -- (201) 547-3094 or 3098.

* * * * * *

Remember, it will be you who will make the difference in preparing students for the future where one day soon computer literacy will be considered a basic skill. By 1990 (or before) everyone here for this meeting will agree with Arthur Luehrmann that "computers are becoming as necessary to a child's formal education as learning to read, write, and do arithmetic." (Luehrmann, Byte, July 1980, p. 98).

APPENDIX

(Resources mentioned on previous pages)

SELECTED RESOURCES FOR DEVELOPING COMPUTER AWARENESS AND FOR TEACHING A COMPUTER LITERACY COURSE. All of the following should be consulted before embarking upon the design of a K-12 Computer Education Program or before launching a Computer Literacy Course or before writing your own course. Undoubtedly, you will end up using all or most of these items in the actual course. You may also not see a need for writing your own course from scratch. Luehrmann and Peckham will have done it for you.

1. Are You Computer Literate? by Karen Billings and David Moursund. Dilithium Press, P.O. Box 92, Forest Grove, Oregon 97116, 1980. 148 pp. $8.95—Note the "Computer Literacy Exam" on pp. 133-139.
2. Minnesota Computer Literacy Questionnaire and Awareness Assessment (Supported by the National Science Foundation, Grant No. SED77-18658), Minnesota Educational Computing Consortium, 2520 Broadway Drive, St. Paul, MN 55113.—Two forms are available: eleventh-grade and eighth-grade.
3. Be A Computer Literate by Marion J. Ball and Sylvia Charp. Creative Computing, 39 East Hanover Ave., Morris Plains, N.J. 07950, 1977. 64 pp. illus. $3.95
4. An Introduction to Computers and Computing by Jean B. Rogers. International Council for Computers in Education, Department of Computer and Information Science, University of Oregon, Eugene, Oregon 97403, 1981. 48 pp. $2.50.—This booklet outlines a course "specificially designed for students at the secondary school level. It is intended to be a general education course to be studied by every student, not restricted to students who are in advanced science and mathematics programs."
5. Apple Seed Program—available in two versions: junior high or senior high—contains six teaching tools.
    a. Computer Literacy Show and Tell Kit
    b. Computer Discovery
    c. Computers and Education
    d. Microcomputer Systems and Apple BASIC
    e. Applesoft Tutorial
    F. Educational Software Directory

Apple Seed includes materials prepared by Apple, Science Research Associates, Inc. (SRA), educational publisher Sterling Swift, and Dr. James S. Poirot, Chairman of the Computer Science Department at North Texas State University in Denton, Texas.

The following is extracted from Apple Education News, Issue 5, January 1981:

COMPUTER LITERACY SHOW AND TELL KIT: Developed by educational publisher Sterling Swift, the kit contains examples of real computer hardware e.g. diodes, floppy disks and integrated circuits. With each piece of hardware is an explanation of its importance to computing. Bound in a durable vinyl notebook, Computer Literacy Show and Tell Kit entertains and informs students and teachers of today's computer developments.

COMPUTER DISCOVERY by Science Research Associates, provides educators comprehensive computer literacy courseware in the form of a 128-page student workbook, instructor's guide, and two Apple diskettes. It covers computer history, hardware and software, analysis and programming, and the social and economic impacts of computers.

Available in junior high or senior high versions, the junior high level is appropriate for gifted or advanced elementary students.

COMPUTERS AND EDUCATION by Dr. James Poirot (89-page softbound) discusses computer history, applications (both in the classroom and in the office), educational games and computer trends. Computers and Education also covers selection of equipment, setting up a computer lab-even creating classroom bulletin board displays.

MICROCOMPUTER SYSTEMS and APPLE BASIC also by Dr. Poirot, (150-page softbound), teaches the computer novice to formulate problems for a computer and then to program with Apple BASIC.

APPLESOFT TUTORIAL by Apple Computer addresses users new to computers and Apple. The Tutorial teaches Apple's capabilities and programming fundamentals. Each school participating in Apple Seed will receive a classroom set (25 copies) of Applesoft Tutorial.

THE EDUCATIONAL SOFTWARE DIRECTORY: Apple II Edition from Sterling Swift Publishing Co. eliminates flipping through catalogues and clipping ads in computer magazines. It is the ONLY comprehensive directory of the educational software available for the Apple Computer System. It includes a short description and price of each piece of software, ordering information, and an index categorizing available lessons by grade level. The Directory covers over 700 separate listings including instructional packages from commercial software publishers as well as non-profit and educational institutions.

Although Apple Seed is no longer available (as a free teaching kit to schools buying their first Apple Computer) its components are available from Sterling Swift, SRA, and Apple.

For example:

The Computer Literacy Show and Tell Kit may be purchased from Sterling Swift for $59.95
Educational Software Directory, Apple II from Sterling Swift, $9.95
Computers and Education from Sterling Swift, $6.95
Microcomputer Systems and Apple BASIC from Sterling Swift, $8.95

Sterling Swift Publishing Co., 1600 Fortview Rd., Austin, Texas 78704 (512) 444-7570

Computer Discovery Program, Junior High or Senior High, Diskette Version is available for Atari (6-3100) or Apple II (6-3101) at $195.00 from Science Research Associates, Inc., 155 North Wacker Drive, Chicago, Illinois 60606. (800) 621-0664. For a review of Computer Discovery, see The Journal of Courseware Review, Vol. 1, No.1, August 1981, pp. 40-43

6. Computer Literacy: Problem Solving with Computers by Carin E. Horn and James L. Poirot. Sterling Swift Publishing Co., 1600 Fortview Road, Austin, Texas 78704, 1981. 304 pp. illus. $13.95

7. Instructional Manual to accompany Computer Literacy: Problem Solving with Computers by Carin E. Horn and James L. Poirot. Manual by Carin E. Horn and Carroll L. Collins. Sterling Swift, 1981. 187 pp. illus. $7.95.--A gold mine of information and ideas for the teacher.

8. Computer Power (Student Text, Program Courseware, and Teacher's Manual). Gregg/McGraw-Hill, 1221 Avenue of the Americas, New York, N.Y. 10020.

9. Radio Shack's TRS-80 Basic Computer Literacy Classroom Package. This consists of a teacher's guide, a wall chart, and six spirit masters for duplicating student activity sheets. The price is $9.95 and it is available at Radio Shack stores, dealers, and Computer Centers.

10. The Computer Literacy Course being developed and field tested by Dr. Arthur Luehrmann and Herbert D. Peckham (co-founders, with Martha Ramírez, of Computer Literacy). Computer Literacy is writing and testing a complete set of curricular materials for a junior high school course in computing. It is a one-semester course based heavily on

Laboratory activities carried out by each student, working individually or in pairs, at a computer. The course is intended for average students and presumes no more than seventh grade math skills. The textbook and lab manuals will be published by McGraw-Hill for use in schools by the Fall of 1982.

The course is built around the lab manual for A FIRST COURSE IN COMPUTING which takes students through carefully structured exercises. In addition to the lab manual there is a text which discusses the material experientially learned in the lab, covers computer awareness topics (applications, ethics, privacy issues, history, implications, etc.) and makes reference to other computer languages, noting their similarities and differences with BASIC.

There are eight test sites around the country that are participating in the field trials this fall. Revisions are expected to be made in February and March, with final publication by McGraw-Hill in May, 1982.

For more information contact McGraw-Hill Book Company, 1221 Avenue of the Americas, New York, N.Y. 10020, or Ms. Martha Ramírez, President, Computer Literacy, 1466 Grizzly Peak Boulevard, Berkeley, CA 94708.

11. Computer Literacy Project (in Grades K-8). Human Resources Research Organization, 300 North Washington Street, Alexandria, Virginia 22314, under a grant from the National Science Foundation. Project Director is Beverly Hunter, Senior Staff Scientist, HumRRO. See "Computer Literacy in Grades K-8" by Beverly Hunter, J. Educational Technology Systems, Vol 10 (1), 1981-82, pp. 59-66. See next page for a reprint relevant to this Project.

MICROCOMPUTERS GO TO SCHOOL

Gary G. Bitter
Department of Elementary Education
(Payne 203)
Arizona State University
Tempe, Arizona 85287

INTRODUCTION

The Micro-electronic revolution of the last decade brought about an unprecedented reduction in the price of microcomputers. The effect has been that of providing highly sophisticated computer hardware within the budgetary limits of most school districts. In general, computers have invaded all our lives in some form or other. Therefore, we must bring awareness and knowledge of this technology to our lives. Schools, teachers, administrators, parents, and chidren have a definite need to become computer literate.

A recent study by the Creative Strategies, Inc., a market research firm, indicates that school purchases of microcomputers will quadruple in the years between 1978 and 1983. Their projections are that units purchased by schools, on a national basis, will grow dramatically from 26,700 in 1978 to 130,000 in 1983. Additionally, 70% of this demand will be from elementary and secondary education. These projections indicate that microcomputers "can" potentially make a major contribution toward improving the present educational delivery system. Current statistics from the National Center for Education indicate that there are 50,000 computer systems in the schools. A breakdown of their information showed that 50% of the high schools and 14% of the elementary schools use at least one microcomputer or terminal for Computer Assisted Instruction (CAI).

Thus, a decade ago few computers were used in elementary and secondary education. Most of them were in the administrative offices performing the necessary functions of administrative tasks such as payrolls and record keeping. Very few were in the classroom for teaching students about computers or how to program them. Now with microcomputers, even young children have the opportunity to interact with a computer. Children from grades one through twelve are learning to program them. And much larger numbers are being exposed to what computers are, what they can and cannot do, and how computers can be useful in helping many to do tasks better and more efficiently.

EDUCATIONAL APPLICATIONS

Microcomputers have many of the same capabilities of the larger mainframe computers and minicomputers. The major difference, of course, is that the larger ones can accomodate larger programs, larger amounts of data, and operate at much faster speeds. The speed of the mainframe is not essential for most classroom purposes and many of the record keeping activities at the school classroom or school site level. The following are the most common educational applications.

Computer-Assisted Instruction

CAI is the acronym for Computer Assisted Instruction. The most common approach is using programs that reproduce lessons of the practice and drill type. These programs are found in almost any subject matter - mathematics, science, reading, spelling, social studies, etc. They enable students to practice material that they may have learned but have not yet mastered. The computer is a far more patient teacher than any human teacher could possibly be. Also, the material is presented in a consistent manner. Some students need the repetition to firmly implant concepts into their mind. Another feature of CAI-type programming is the ability to teach the student only the material that has not been mastered. The material already mastered can be skipped over, saving the student time and avoiding boredom from going over material already learned.

Essentially, using a computer for drill and practice enables the computer to make a determination on the student's ability to learn new material or to master previously presented material. Based on the ability of the student, the program will branch to the routine(s) that will enable the student to receive help as needed in the amounts and types needed. With immediate feedback on one's own progress, the student is enabled to learn material that was missed in class or simply in the earlier portions of the computer program. Once again the computer is the patient, tolerant, nontiring teacher, that no human could ever hope to duplicate. No computer can replace a teacher, but computers can be of definite assistance to a teacher that has more students that need to be tutored than there is time to do an adequate job of individual tutoring.

Another role of the computer in education is through tutorial programs. Tutorial programs are characterized by the ability to teach concepts; for example, to teach how the heart works. While practice and drill programs are good for mastering material, they may have few tutorial or teaching techniques built into them.

Graphics and sound capabilities of microcomputers make tutorial teaching via a computer a much more viable academic endeavor. Through carefully sequenced instruction and illustration, people of all ages can be taught concepts using the computer. Of course, this type of instruction is very expensive to develop and is presently quite limited in availability. Another feature of CAI tutorial programs is that the students are able to go at their own pace. The student is competing only with oneself and not with others. This obviously can boost one's self-awareness.

The computer can be used to simulate events in the world. This can be very useful and necessary for several reasons. Some concepts might be too dangerous for the students, teachers, or anyone else to get involved with directly. For example, one cannot expect everyone to get into an airplane and fly it without going through a lot of training. With simulation landing programs, one can land a Boeing 747 on a runway on the microcomputer. Crashing the plane load of passengers might flatten one's own ego, but certainly no harm would come to the fictional passengers or crew. Another would be a simulation of a nuclear plant accident. The whole scenario could be done safely using a microcomputer. The students learn firsthand what happens, given a set of circumstances, without putting oneself or the general public's health in jeopardy. Some concepts are difficult to teach because of time loss. If one wanted to study the effects of genetics over several generations of humans, that would be an impossible task. By simulating the conditions in a computer, the results can be projected with a much clearer understanding of the processes involved. We can take a look into the future without actually being there, and perhaps in the process prevent potential dangers that might not otherwise be possible without the computer simulation. Another interesting application of computer simulation is the famous "lemonade" stand. Here, children can learn the concepts of economics at a level that they can understand, yet producing realistic consequences of the economic system. This duplication of an economic environment clearly would be impractical in the real world and actual solutions, as well as any predictions that would be generated. A computer may be programmed to calculate a loan. This is another example of CAI called problem solving. In this way, the computer can be used as a decision facilitator to determine whether or not to buy or rent a home. Having students write programs on math, science, social sciences, etc., is an effective use of the computer. In other words, programming the computer requires the student to understand the problem in order to get the computer to carry out the problem solving detail.

Games

Games are another computer use and can be for pure enjoyment, but educational value can also be gained with some computer games. Many students, especially learning-disabled students, need work in developing eye/hand coordination. Many games have this feature. This is especially true of the arcade-type games like pinball, "star wars", racing games, flight landing simulators, etc. If the games have graduated-ability levels, the youngsters (and some adults) can improve their coordination while thinking they are having fun. Strategy games require the person to apply logical reasoning for solutions. Some common games are known to all of us, like checkers or chess. Some versions will match the player against the user, while others provide choices of playing against another human opponent or the computer. Different levels of difficulty programmed into the game add educational challenges as well as personal enjoyment.

Administrative Applications

Administrative uses of computers occur both in the classroom and in administrative offices. The administrative offices have used computers for many years, especially in the larger districts from high school through college. One use of computers by the classroom teacher is for computer managed instruction. This is particularly true of teachers that have students in special education. Here teachers must prepare an Individualized Education Program (IEP) required by Federal law. The program is often times done by hand. Where it is done by computer, there is a much greater savings of time and effort on the teacher's part. Also, with a well planned IEP program to manage the instructional plans, the teacher is more apt to provide the right materials and programs that truly would make for the best program for each student. Other aspects of different computer management instruction sequences are simply a way of keeping track of the student's progress within a particular curriculum run on a computer system. The computer keeps track of the student's prograss at all times and prepares reports for the teacher, indicating strengths and weaknesses. Thus, teachers have accurate records at all times enabling the instructor to plan appropriately for the student's progress.

The second administrative function is one shared both by the classroom and school office. Though the programs may be different, the idea is the same - student record keeping. Every teacher must keep track of the student's grades in the classroom. In regular classrooms, most teachers have 30 or more students to keep track of in many different subject areas. With a computerized gradebook, the teacher has immediate

71

access and knowledge of every student's progress with the touch of a few key strokes on the computer. If a parent wants an immediate report of the progress, the teacher does not need to go into panic and spend a half hour just to calculate all the scores for one student. The computer gradebook can provide this information immediately. It can be a tremendous timesaver for any teacher. What's more, valuable information not otherwise available can be obtained with a good gradebook software package including class rankings, the scatter of grades, averages, test validity, etc. From the administrative side, there are attendance reports which need to be kept daily. They are not only time-consuming but tedious. Also, absentee reports can more easily identify the chronically absent student. Finally, the administrative offices at the school building, school district, and higher-education levels have all the usual administrative tasks of running schools. Budgets may be prepared, expenditures recorded, salaries paid, Federal and State reports prepared, etc., etc.

Needs Assessment

In the Spring of 1980, a survey of the needs of the school educators of Arizona was initiated by the microcomputer study group of the College of Education at Arizona State University. The computer-assisted instruction needs assessment study consisted of a questionaire which was sent to every elementary and secondary school district in Arizona. Forty-six districts responded to this questionaire. The study indicated that 37 of the responding districts were using microcomputers for instructional purposes. Fourteen more had future plans of utilizing microcomputers, but only a small percentage of students were actually involved in microcomputer-assisted instruction. The conclusion of the needs assessment follows.

- Software Availability: Seventy percent of all respondents indicated a desire to exchange information and software.

- Software Development: Sixty-five percent of all responding districts are interested in participating in cooperative software development projects.

- Teacher Training: Sixty-five percent of all districts that have or plan to have computer activities are interested in teacher-training workshops.

In response to the educational needs assessment survey, the College of Education initiates and supports several computer-related activities. The following is a summary of each activity.

THE MICROCOMPUTER LABORATORY

The College of Education Microcomputer Laboratory was developed in July 1980. Five microcomputers (now six) were placed in the Instructional Resource Laboratory for use by faculty and students in the educational programs, as well as being available to the public. The six computers are representative of education's most popular micros today. They are the APF, Apple, ATARI, Commodore PET, Radio Shack TRS-80, and Texas Instruments 99/4 with LOGO.

The Goals of the Microcomputer Laboratory are to:

- Provide support for courses already offered in the COE.
- Furnish opportunities for faculty to become familiar with characteristics of many different types of microcomputers.
- Be utilized for microcomputer research.
- Benefit the educational community.
- Act as a clearinghouse for computer programs.

The Microcomputer Laboratory is open to the general public on Tuesday, from 5 to 8 p.m. A graduate assistant is available during this time to assist the user. This laboratory gives teachers, administrators, and others an opportunity to preview the most common machines as well as to review much of the current software. Each of the computer systems have a growing library of software as well as a subject or topic directory of what is available. Most of the current software includes the academic areas of reading, math, science, social sciences, and spelling. Included in the library are simulations, drill and practice, and tutorial and game software. Brief teacher evaluations are available for the majority of software. In addition, the microcomputer laboratory can be reserved for teacher and administrator meetings, College of Education classes, and research activities. Several COE classes regularly use the facility.

TEACHING TEACHERS ABOUT MICROCOMPUTERS

Teaching teachers about microcomputers is another activity supported by the COE. Through a grant from the National Science Foundation, a Computer Literacy project was started in the Summer of 1981. This program will continue through 1982 in the following three phases.

Phase I of the project involved school administrators from participating districts. This computer-literacy awareness session concentrated on computer literacy, introduction to microcomputers, discussions of opportunities for improvement of the educational delivery system brought about by microcomputer technology, and a summary of the most commonly used educational hardware. Also, there were hands-on demonstrations. In addition, the administrators helped to organize computer-literacy courses in their own district.

The second phase of the project was a four-week intensive study for 30 participants. These 30 participants are teachers of science education in grades 5 through 12. This study included general computer-literacy topics, different programming languages used with microcomputers in education, learning to program hardware, and software evaluation.

Ten teams were established to direct computer-literacy classes as part of Phase III. Each team developed a detailed computer-literacy class which will be offered during the 1981-82 academic year at ten different school districts. Each class will meet for 18 contact hours and provide district professional growth credit on their salary scale. The topics covered will include computer literacy and introduction to microcomputers, the promise computers hold for improving effectiveness of instruction, introduction to BASIC, and hands-on experience for hardware/software evaluation.

The Phase II participants will teach the classes in their local participating districts in cooperation with faculty from the College of Education. Twelve classes are scheduled for the academic year. (Call 965-3322 for an 81-82 schedule.)

MICROCOMPUTER CONFERENCE

The College of Education conducted a microcomputer conference in January of 1980. The goal of "The Microcomputers in Education: Getting Started" conference was to introduce elementary and secondary teachers and administrators to microcomputers. The theme emphasized elementary education, secondary education, special education, administrative, and general interest sessions. The main speakers told about the challenge of the 80's: computer literacy, computer-assisted instruction in the basic skill areas at the elementary and secondary levels; applications for microcomputers in special and gifted education; evaluating microcomputer systems; hardware and software comparisons; career education and guidance information systems; instructional techniques for teaching the

BASIC computer programming language to elementary and secondary students; designing computer proposals for Federal Funding; and microcomputers in the fine arts areas. The conference was attended by over 400 individuals. There were 55 speakers and 18 exhibitors.

COMPUTER EDUCATION CLASSES

The College of Education offers several classes on or related to computers. These classes emphasize computer literacy, computer-assisted instruction and/or programming. More courses are planned for the future. Topics covered will be PILOT, LOGO, word processing, graphics programming, advanced BASIC programming and techniques of computer use including simulations, problem solving, drill and practice, and tutorials. In addition, several classes incorporate the computer into the traditional class. These classes are regular offerings in Educational Psychology, Higher Education, Educational Technology, and Elementary Education.

CONCLUSION

This is a summary of the major commitments of the College of Education at Arizona State University to microcomputers in education. These avenues provide educators a start in utilizing the microcomputer in education. The microelectronic revolution is still in its infant stages and educators need to be aware of its strengths and weaknesses. If we can research and understand its implications, the future of microcomputers in education can be rewarding. Our charge is to determine the most educationally sound approach to harness this accessible and purposeful capability.

BIBLIOGRAPHY

Akers, Joan, et.al. "Classrooms Make Friends With Computers," Instructor, February 1980, pp. 52-58.

Barstow, Daniel. "Computers and Education: Some Questions of Values," Creative Computing, February, 1979, pp. 116-119.

Bitter, Gary G. "Computers and Computer Resources," TEACHER, 97 (5): 62-63, 1980 "Computer Literacy Needs," Microcomputers in Education, Arizona State University, Tempe, AZ, 1981, p. 1:10. Exploring With Computers, Julian Messner, New York, 1981. Microcomputer Applications for Calculus, Prindle, Weber andSchmidt, Boston, 1972, Revision, 1982.

Carpenter, Thomas, et.al. "The Current Status of Computer Literacy," The Mathematics Teacher, December 1980, pp. 669-673.

Jay, Dr. Timothy B. "Are You a Computer Literate," The Computing Teacher, Volume 8, No. 4, Academic Year 1980-1981, pp. 58-59.

Jones, Beverly. "Computers in the Arts and Humanities," The Computing Teacher, Volume 8, No. 6, Academic Year 1980-1981, pp. 22-29.

Mazur, Ken. "Grow Old Along With Me! The Best is Yet to Be..," Creative Computing, August 1980, pp. 39-42.

Molnar, Andrew. "The Coming of Computer Literacy: Are We Prepared for It?" Educational Technology, January, 1981, pp. 26-28.

National Council of Teachers of Mathematics. "An Agenda for Action," Reston, VA 1980.

Olds, Henry F. Jr. "The Making of Software," Classroom Computer News, July-August 1981, pp. 20-21.

"Show Plugs Home Computers," Arizona Republic, July 25, 1981, p. c3.

Survey of Arizona Public School Practices and Needs for Computer Assisted Instruction, College of Education, Arizona State University, Tempe, AZ 85287, 1980.

****

Dr. Gary G. Bitter, professor of Mathematics and Computer Education in ASU's College of Education, is a consultant to several microcomputer projects and author of numerous articles on computers. He is editor of The School Science and Mathematics Journal.

ONE SOLUTION TO THE COURSEWARE CRISIS

by

Barbara R. Sadowski
University of Houston
Department of Curriculum & Instruction
Houston, Texas 77004

At the 1981 Microcomputers in Education Conference held in Arizona State University, Dr. Judith Edwards, the featured luncheon speaker, spoke on "The Courseware Crisis: Help is on the Way." In her talk, Dr. Edwards stated a crucial obstacle to the wider use of the microcomputer: lack of adequate materials including a lack of documentation and dissemination of the materials. This is not news to any educator who has tried to acquire software that is educationally sound, relevant to their district's needs, and available at a reasonable price. Horror stories abound at conferences about experiences with software that does not live up to its promotional advertising which almost always promises much more than the disk delivers. Some school districts have ordered software in good faith only to throw out over half the disks after teachers see it once. Other school district administrators have become extremely wary about purchasing software and only purchase after a preview by their teachers or upon recommendation from other users.

As more and more software firms spring up, the problem only increases, with no easy solution in sight. Granted, there are attempts being made at the national level to solve this problem; the Northwest Regional Educational Laboratory's project MicroSIFT, funded by the National Institute of Education is one such attempt. MicroSIFT (Microcomputer Software and Information for Teachers) is a clearinghouse for microcomputer instructional software and teacher information which has established effective procedures for the collection, evaluation, and dissemination of materials, and information about microcomputer software packages and materials for educational use at K-12 levels. The evaluation model being used has four stages: Sifting, Description, Evaluation, and In-Depth Evaluation. A software package that passes the first three stages is assessed by its ultimate consumer—the student. This project has been in the process of developing and validating its evaluation model for over a year, but its impact on software developers has yet to be felt.

CONDUIT is another organization which reviews, tests, packages, and distributes instructional materials for the computer, primarily for instruction at the postsecondary level. Other software clearinghouses, both commercial and public, attempt to serve a national audience of consumers, and while the effort is commendable, it is not very efficient for decisions made at the local level. Because of the explosion in the number of software producers, the national clearinghouses do not have the resources to evaluate every software package that a local school district might be interested in purchasing. And even if they were able to provide a review for every available commercial package, other problems remain.

For some school districts, price is often an obstacle to acquiring quality software, although disk leasing can be a viable alternagive. Another difficulty with commercial software is its instructional inflexibility. For example, a school might want to modify a definition in an instructional program, change the sequence or timing of a drill and practice lesson, turn off the sound on a program, or even correct a bug in the program. Much of the time, these minor modifications are not possible because the program is locked to protect the copyright, as it rightfully should be to safeguard the author's rights. As a result, the program listing is not accessible, and the teacher must use the program as is with the undesirable features or not use it at all. Ideally, software purchased for instruction should be as flexible as a workbook, so that teachers can modify or eliminate parts they don't want and add features they do. However, this seems to be a dream for the future.

With all of the problems listed above, it is not surprising that there is a software crisis, a crisis that may be big enough to set back the educational use of microcomputers by several years. The solution for some schools is to find a teacher who knows not only how to write computer programs but also knows what can be taught effectively on the computer. By utilizing this teacher's skills in programming and instruction, a school can develop its own grade book program for computer-managed instruction and instructional programs tailored to meet the needs of the teachers. The major drawback to this solution, of course, is the time that it takes for one teacher working alone to produce high quality software. When two such computer resource teachers meet they often discover that they've developed similar sets of programs, since teachers' needs for computer-enhanced instruction are very often the same.

This wasteful duplication of effort in software development could be avoided if there existed some means of coordination between schools and teachers. In Houston, Texas, two agencies working together provide such coordination for teacher-developed software.

One agency is the Region IV Education Services Center (Region IV ESC), and the other is the University of Houston College of Education (UH). The Region IV ESC first coordinated the efforts of its 56 school districts in writing a contract for 600 microcomputers in 1979. This contract, which was, at that time, the largest purchase of computers for educational purposes in the U.S., was awarded to Bell and Howell for the Apple II Plus 48K microcomputer. In 1981 this contract was extended for another 200 Apple microcomputers. The Houston Independent School District (HISD), the largest school district in Region IV with about 190,000 students and 10,000 teachers made the largest purchase of these microcomputers. The ESC staff conducts three-day computer orientation inservice sessions for teachers in school districts that purchase microcomputers under the ESC contract.

78

Thus, by early 1980, there were area teachers who had access to microcomputers, had received some computer awareness, and wanted to know more about programming for instruction. A graduate course in BASIC programming for instructional purposes offered by the University of Houston met this need.

This course focused on learning enough programming in BASIC to write instructional programs. The major assignment for each teacher was to write a program which would be useful for them in their classroom. The first time this course was offered, projects were written on topics such as (1) drill and practice on the basic facts, integers, simplifying radicals, and reducing fractions; (2) tutorials on the Pythagorean theorem, order of operations, and multiplying binomials; (3) games on place value concepts, a version of BAGELS, and most importantly; (4) a tutorial with a timed practice on number sense problems. This last program was of interest to several teachers who were advisors to the Number Sense Club in their schools. Each teacher received a hard copy listing of every project developed in the class.

The next semester, teachers in the course had a choice when deciding what to do for their project. They could modify an existing program, extend the content or scope of an existing program, or author a new program to meet their classroom needs. One student elected to extend the number sense program to cover more topics, then wrote a program which randomly generated items for number sense practice tests. Other students developed games to practice math skills and tutorials such as the one which provides instruction on solving linear equations followed by a drill and practice on a set of randomly generated linear equations at four difficulty levels. Each teacher again received a copy of each program in the class and a copy of the programs from the previous semester.

This process of sharing, modifying, and extending teacher-developed programs has continued now for almost two years. Some programs like the number sense test program have outgrown the 48K RAM of the Apple II. The version that generated both the test items and the teacher's answer key was the straw that broke the Apple's back, so now there are different versions of this program for elementary, junior high, and high school number sense tests. The librarians and curriculum supervisors in the course have written information management programs which keep track of materials and books while another program provides a test on basic library skills. Science teachers have written games, tutorials (on the gas laws), simulations (for genetics), and drill and practice on classifying animals. The most impressive program is a science game called LOST IN SPACE which gives the student 100 energy units to expend on a trip to various bodies in the solar system. You land on an unknown body, and by conducting experiments which cost you energy units, you must determine your location. The object is to land on and identify as many locations as you can before all your energy units are expended. Students and teachers love to play this again and again.

Programs to aid the teacher in mathematics at every grade level have been developed. Some programs help the teacher in keeping track of student grades, others print out pages of factorable quadratic equations or pairs of binomials to be multiplied. Drill and practice on almost everything possible have been developed as well as tutorials and games. One program converts Roman numerals to Hindu-Arabic decimal numerals and vice versa and even lets the student practice converting numerals. A mental math program which controls the rate at which numerals appear on the screen is a very popular one with many teachers. A diagnostic test, tutorial, and drill and practice on the LCM is one program which has been extended by three different students.

Tutorials and tests have been written about computer literacy skills and programming the computer. A bicycle safety instructional program and a program for beginning students in drafting each include graphics, although this topic is not covered in the first course. Students in the course the last semester wrote several programs on geometry concepts, place value concepts and a superb spelling practice program, along with a drill and practice on multiplication and division by powers of ten.

The programs described above vary considerably in complexity and content as might be expected from teacher-developed software. However, most are personalized, menu driven, student prompted, have an appropriate number of trials, provide positive and encouraging feedback with readability levels that match the grade-level content. The lack of sophistication is offset by two important factors. One is that the programs are free, and secondly, teachers can modify the program to serve their own needs. Further, because local teachers have developed the programs to meet their specific needs, other local teachers find that the programs also meet their needs. This last point is an important reason for the success of the software sharing network.

Dissemination of programs within the network is provided by a disk library of all teacher-developed non-proprietary programs at the Region IV ESC which is available for teachers to copy. Teachers are also free to come to the College of Education microcomputer lab at the University of Houston to copy programs from disk or run hard copy listings. Inservice workshops demonstrating the teacher-developed programs also alert teachers to the library of educational software at the University of Houston and Region IV ESC. Presentations at state and national conferences have resulted in requests for listings and disks.

While the coordination and sharing of teacher-developed programs may not be the ultimate solution for the software crisis, it seems to be working well for schools and teachers in Houston. The advantages of little or no cost, accessibility to the listing for program modification, and teacher development to meet local needs are hard to beat. The coordination of a software sharing network of teacher-developed programs requires some central person or agency that has regular contact with teachers interested in using computers for instruction. The payoff in free software is well worth the time and effort involved in establishing such a network. As one popular poster put it: "No one of us is as smart as all of us," and so, by working together, the courseware crisis may be solved.

II. RESEARCH AND PROJECTS

BASIC SKILLS THROUGH MICROCOMPUTERS

Kay Gilliland
Emery Unified School District
4727 San Pablo Avenue
Emeryville, California 94608

All available resources must be utilized to improve instruction so that all students are enabled to master the basic skills. Expanded use of technology may prove to be effective in meeting the needs of all students. The use of microcomputer programs to supplement classroom instruction may help teachers solve one of our most pressing educational problems.

The computer originated as an expensive tool of industry, a basic element of space-age technology. Demand grew, prices dropped, and microcomputers appeared in the homes of affluent students. If the educational potential of the microcomputer is not to become the prerogative of the advantaged student, computers must be available in the schools. If all children are to benefit from this powerful new technology, ways must be found to integrate the use of the computer into the school curriculum.

One way is to use the computer to supplement classroom instruction through drill and practice. It is of importance to educators to know whether or not such use contributes to students' attainment of skills.

Two national organizations which have expressed their vital interest in mathematics education are the National Council of Teachers of Mathematics and the Conference Board of the Mathematical Sciences. The National Council of Teachers of Mathematics developed a position paper, An Agenda for Action--Recommendations for School Mathematics of the 1980s. The Conference Board of the Mathematical Sciences appointed a National Advisory Committee on Mathematical Education. NCTM stressed the importance of the use of computers in mathematics education at all grade levels, while the NACOME Report emphasized the need for continued research on the use of computers in mathematics education.

The Basic Skills Through Microcomputers Project evaluated the effects of the use of microcomputers in classroom instruction. Thus it used the power of computers as recommended by NCTM and provided the research urged in the NACOME Report.

This study assessed the effectiveness of the use of microcomputers to supplement classroom instruction in mathematics, language, and reading for the children in the fourth, fifth, and sixth grades in regular classes in the public school system of a low-income area. (The Emery Unified School District is located in a low-income urban-industrial area serving a multi-ethnic community. It has the highest minority enrollment of any school district in the county. The student population is 69% Black, 14% Hispanic, 6% other ethnic minority, for a total minority student enrollment of 89%.)

The available studies of the effectiveness of computer-assisted instruction have focused on work with large computer systems. In a survey of this research, Marie Dence (1980) found that CAI was more effective than traditional instruction in 45% of the studies surveyed. It was at least as effective in 55%. Alex Dunn (1974) reported on a three-year study of sixth-graders using CAI in the Montgomery City Public Schools. The students using computers made significantly greater gains. These gains were retained over the summer. In the Elementary Mathematics Project, conducted by the Computer-Based Educational Research Laboratory at the University of Illinois in conjunction with Champaign-Urbana Public Schools, experimental results showed that children in the fourth through sixth grades using CAI on the PLATO computer system showed positive gains in mathematics relative to control groups.

Although the National Center for Educational Statistics reports that the majority of computers now used in the schools are micros, very little research has been produced. In "Classroom Computers, Do They Make a Difference?", Spencer and Baskin (1981) state, "...the lack of research is no surprise: the first microcomputer was marketed in 1975, the first easy-to-use version appeared in 1977, and even as late as 1979, most microcomputers were purchased for business rather than instructional use." Billings comments, "As long as small businesses are purchasing more microcomputers than schools, the hardware and software developers will direct their efforts in that direction." (Billings, 1980). Research follows use; as usage increases, research will increase.

Support for research on the educational use of computers was emphasized in the NACOME Report: "Research is urgently needed concerning the uses of computing and calculating instruments in curriculum at all levels and their relationship to a broad array of instructional objectives."

Priorities in School Mathematics, the PRISM study (1980), gave strong support for the use of computers in mathematics. Nearly 75% of the professionals sampled and 80% of the lay people sampled believed that the use of computers and other technology in mathematics classrooms should be increased during the 1980s. About 58% of the respondents supported student interaction with computers in the primary grades and a larger number were in favor of computer use by students in the elementary grades.

In An Agenda for Action: Recommendations for School Mathematics of the 1980s, the NCTM recommends that "...mathematics programs take full advantage of the power of computers and calculators at all grade levels." NCTM makes it clear that "it is imperative that schools play an active part in preparing students of the 1980s to live in a world in which more and more functions are being performed by computers." The first recommendation of NCTM in this regard is, "All students should have access to calculators and increasingly to computers throughout their school mathematics program. Schools should provide calculators and computers for use in elementary and secondary classrooms."

The computer programs used in this study are intended to supplement classroom instruction. They are designed for drill and practice rather than for tutoring. Many of the programs used in this study are games and simulations which require problem-solving, spatial concept formation, estimation, and decision-making. These aspects of the project may be very beneficial educationally, but they were not considered in this study.

Students who are exposed to computer technology and who use a computer as an accepted tool in their school work may be better-equipped to cope with a world where jobs and everyday experience increasingly depend on computer literacy. However, this study was not intended to assess such changes.

The Comprehensive Test of Basic Skills was given to all students in the fourth, fifth and sixth grades in October, 1980, before the Computer Center opened. The results provided the base-line data to be used in evaluation of the effect of the project on student achievement in the basic skills of mathematics. The CTBS was given again in May, 1981, and in October, 1981, to provide post-test information on all of the above students.

The Computer Center opened in October, 1980, after the pre-testing had been completed. The Center had twelve microcomputers, a disc system and a printer. Students came with their teachers and/or classroom aides four times each week. They were also allowed to use the Center on their own time at lunch and after school.

Three staff members worked with the Basic Skills Through Microcomputers project. The Project Director was responsible for the overall management of the project, orientation to district policies and procedures, coordination with the Superintendent and Board of Trustees, development of the budget and cost-accounting system, and planning for the continuation of the project. The Project Coordinator was responsible for site management of the project, orientation to school policies and procedures, and coordination with the principal and teachers involved in the project. The Computer Specialist was responsible for operation of the computer facility during school hours and for drop-in student visits during lunch and after school. She was responsible for interaction with the teachers on a daily basis concerning program requests, modification of programs to meet the requests, and changes which are needed to meet individual class needs.

The Emery Unified School District provided a room with full utilities, furnishings and custodial service at Ralph Hawley School where teachers brought their classes to use the computers. The Computer Center is located on the school grounds with its own outside entrance, making it easily accessible both from the classrooms and from the school grounds. Security is provided by an alarm system and by a special fastening arrangement for the computers. The Computer Center was in use by students and/or staff from 8:00 AM until 4:00 PM every school day during the study.

A thorough investigation of the types and models of microcomputers which were available resulted in the purchase of PET computer equipment designed and built by Commodore Business Machines, Inc. The PET equipment was chosen because it had a history of reliability in school use, there was more educational software currently available for the PET, and the PET computers were considerably less expensive than other systems. The following equipment was purchased, set up, and in operation by September 5, 1980:

EQUIPMENT	QTY.
PET 2001 Professional Computer	12
CBM 2022 Printer	1
CBM 2040 Dual Drive Floppy Disk	1
CBM C2N Cassette Drive	6
Cable	12
Cable	4

One computer was purchased with a tool kit and one was upgraded to 32K.

Modifications have been made in the equipment and the room arrangement during the period of this study. It was found that significant oral communication between two students was difficult when the tables were crowded together to the extent they were. The tables were moved slightly and some improvement was noticed. Eventually, the one metre cables were traded for two metre cables and the tables were moved apart to a greater degree. To move them further apart would require purchase of an additional disk drive, due to the electrical limitations of the equipment. Equipment down-time has been carefully monitored. For example, during the period from October 6, 1980 to January 16, 1981, the average of computers in working order was 11.37 out of 12.

A thorough search of the available computer software was made. Although some programs were purchased by Basic Skills Through Microcomputers, many were given to the project. Software has been obtained from:

San Francisco State University, California
Lawrence Hall of Science, Univ. of Calif.
Computer Using Educators, California
San Mateo County, CA, Office of Education
Microcomputers in Education, Minnesota
P C Computers, El Cerrito, California

Software evaluation and development was an on-going process.

Decisions concerning the Computer Center curriculum were made by the teachers for their individual classes. The system for input from teachers was set up to give maximum flexibility for integration of the computer lessons into the regular instructional program. Careful planning and advance inservice ensured that every teacher would have regular input from the beginning of the program. Consistent effort on the part of the teachers and the project staff made it possible for the computer curriculum to reflect the classroom instructional program. Every teacher, every week during the project, submitted a Program Request Form. Through this communication, the teacher requested particular programs and detailed extensive revisions of those programs to meet the needs of his or her individual class.

The result of this effort is best shown by comparing percentile rankings of students over a full year period. Unfortunately, percentile scores can be compared for only one group, the fifth grade students.

The fourth and sixth grade students took different forms of the test in 1980 from those taken in 1981. The fifth graders, however, showed remarkable progress.

SUBJECT	OCTOBER, 1980	OCTOBER, 1981
reading	24th percentile	46th percentile
language	31st percentile	52nd percentile
mathematics	18th percentile	38th percentile

The study is continuing into its second year. It is expected that students will continue to improve their grasp of basic skills through the use of microcomputers.

DATA ACQUISITION, DATA HANDLING CONSIDERATIONS USING MINICOMPUTERS

Pauline Fortier Briggs
Digital Equipment Corporation
1 Iron Way
Marlboro, Massachusetts 01752

In acquiring data in a laboratory from an analytical instrument, many factors must be considered. Among the primary data handling considerations are:

1. Sampling Rate
2. Mass Storage

The Nyquist Frequency sampling theory, states that sampling rate must be at least twice the frequency of the highest variable frequency of interest. Sampling of a signal below this threshold rate will result in distortion of the waveform to produce an apparent signal of lower frequency (and possibly lower amplitude) than the actual signal. The distortion observed in both cases is called "Aliasing" Aliasing occurs whenever the sampling rate is less than the signal frequency. The frequency at which distortion or aliasing begins is called the Nyquist frequency. Sampling less than twice the frequency of interest produces loss of resolution and/or frequency information.

Next in our data handling considerations is mass storage. Mass storage is a function of sampling duration and frequency. High resolution or long-term sampling durations require moderate to large amounts of mass storage.

A couple examples of disk - sizing examples are that of high pressure liquid chromatography and nuclear magnetic resolution.

DISK SIZING EXAMPLES

1. HPLC

   Detector at 0.02 AUFS capable of measuring absorbances down to 0.001

   Assuming the absorbances are stored as real numbers requiring 4 bytes, at 25 samples/sec for a total of 30 minutes, one sample will require 180KB of disk storage.

   A floppy based system will only store 5 chromatograms. Such applications as PTH - amino acids and capillary GC will easily require such mass storage.

2. IR

If a researcher chooses a high resolution infrared spectrophotometer it is because he is aware that the nature of his experiment requires such high resolution. Modern IR spectrophotometers routinely have resolution specifications better than 1.0cm^{-1}. The researcher does not want to loose the high resolution of his IR because of the sampling rate of his computer. Sampling at 1.0cm^{-1} between 4000 and 250cm^{-1} requires 15KB of disk space.

Sampling at 1.0cm^{-1} however, because of the Nyquist frequency sampling theory, yields a resolution of 4-6cm^{-1}. The resolution of the IR is degraded by the computer sampling rate.

Of utmost importance in laboratory computers is S/W, it is even more important than the hardware. S/W allows you to achieve your research goals, giving you flexibility to adjust to your changing research and expandability to meet your growing research needs.

Software can be divided in two (2) categories, operating systems and applications languages. An operating system is a set of programs that manage computer resources to provide efficient computer operation. Operating Systems provide services for application program development and act as an environment in which application programs run. The executive of the operating system is the control component and is responsible for the coordination, control and organization of the computer.

The device handlers are the interface between the system and any peripheral device. This allows various input/output devices or equipment to communicate with the computer. This allows interfacing to laboratory equipment in real-time mode, as well as allowing people in a time sharing mode to utilize the computer's resources.

Among the time sharing applications are education and business. Software which are time sharing operating systems and which are completely people oriented are RSTS/E. Among the real-time applications are industrial control, science and engineering. Operating systems which are machine oriented are RT-11. Operating systems which will support both machines and people, intermediate in real-time sharing and real-time capabilities such as; VMS in Digital's VAX, RSX-11M in Digital's popular PDP-11 Family of Minicomputers.

Another group of software is in the class of applications languages. Among the most widely used computer high level languages today are BASIC for the novice computer user. BASIC is easy to learn and use and is usually interactive with the programmer. FORTRAN is for the more sophisticated programmer. FORTRAN allows for easy translation of mathematical equations and efficient execution of numerical calculations.

Popular today among the high level languages is PASCAL, also for the novice user. PASCAL is a structured language which is easy to learn, read and maintain. For the highly sophisticated programmer there is assembly language, used in science and engineering. Assembly language, MACRO in Digital's computers allows for precise control of hardware, faster execution and uses less computer resources.

Application Languages when properly assembled and written, result in applications programs. Applications programs can be of two (2) natures; prewritten "canned" software and optimizable software. Canned software is good for routine analysis or repetitive analysis such as in a quality control lab. These programs however, do not allow for any future growth or expandility as new applications or needs arise. Computers with dedicated applications programs can also only be used for only one (1) application or analytical technique.

Optimizable software is excellent for analysis that demand flexibility and expandibility. Optimizable software consists of prewritten applications subroutines that can be organized by a few simple commands.

Digital Equipment Corporation's Laboratory Computer is the MINC, which stands for Modular Instrument Computer. The MINC allows for easy interface to any instrumentation whether it have an analog, digital, serial that is RS232, or IEEE interface.

With the MINC, there are many prewritten applications subroutines which can be used in laboratory applications programs. There are laboratory subroutine package algorithms which a-low data acquisition and processing. There are scientific subroutine packages algorithms which allow analysis of data.

In conclusion, computers in research have enabled the scientist to expand the range and complexity of experiments and research. By using computers, experiments not too long ago considered impossible or tedious are now routine.

EARLY LEARNING OF LOGIC AND GEOMETRY USING MICROCOMPUTERS

TERI HOCH PERL & ANN M. PIESTRUP
ADVANCED LEARNING TECHNOLOGY, INC.

Under an NSF/NIE grant, ALT has developed microcomputer software that uses geometric shapes and logic elements to develop logical thinking skills. The programs display colorful animated graphics; the environment is interactive; players use a joystick input device to control the flow of learning.

Programs developed under the grant include 1) LOGIC GATES in which children build simulations of machines using "working" logic elements, 2) LOGIC PUZZLES in which children move through rooms solving attribute puzzles and "guess my rule" games, and 3) TURN AND MATCH WORLD in which players reorder a world by carrying rotated objects in search of "matching" rooms.

All materials were pilot tested with sixteen gifted second and third grade students, in two Palo Alto schools, during the 1980-81 school year. Students worked in pairs, over a four week period, forty-five minutes at each session. The programs were evaluated for ease of use, student's ability to work with logic concepts, and student attitudes. Subsequent testing is establishing a wider age range for the use of these materials.

The results of the pilot test show that children enjoyed working with logic concepts using these programs. Students worked well in pairs. Parents, teachers and principal considered the experience so positive that parent volunteers arranged additional hours for more children to share the experience. Results also suggest that informal settings may be a better environment than schools, for this type of exploratory learning. Forty-five minute sessions seemed too short for the interest that was generated.

Programs developed under the grant will be available soon. For details write to:

Advanced Learning Technology, Inc.
4370 Alpine Road
Portola Valley, CA 94025

93

HERE COME THE TURTLES!
Microcomputers in Education, from the Young Peoples' Perspective.

by James H. Muller
Young Peoples' LOGO Association
1208 Hillsdale Drive, Richardson, Texas 75081

POINTS TO PONDER*

> "And this is the way to educate children: the instinctive way of mothers. There should be no effort made to teach children to think, to have ideas. Only to lift them and urge them into dynamic activity. The voice of dynamic sound, not the words of understanding. Damn understanding. Gestures, and touch, and expression of face, not theory. Never have ideas about children -- and never have ideas *for* them."

Reading through the "Points to Ponder" column of the December 1981 issue of Reader's Digest, this quote from D. H. Lawrence jumped off the page at me. That's what the Young Peoples' LOGO Association is all about.

It all seemed so simple at first -- back during the summer -- buy a few LOGO systems and let our son, Larry, and his friends, enjoy the fun of a young peoples' users' group and software exchange. Couple this with a newsletter program and that would be something of a first.

We had tried organizing or joining other users' groups but these were all too adult- oriented. Both Larry and I had also taken programming courses through a local community college program. But these courses were too short, they did not give either of us adequate time on the computer, and they were too limited in scope.

On top of these limitations of the courses, Larry had to wait several months for the next level of his course. And to go beyond that twelve hours, he had to wait until he was an adult. Colleges and other courses did not allow young people to take the advanced courses.

Another important element of the YPLA evolved concurrently with our interest in a young peoples' users' group. I worked with a remarkable paraplegic gentleman from Fort Worth who was developing an entry into the Johns Hopkins University First Annual Search for Personal Computer Applications for the Handicapped. I was also exploring the possibilities of developing audio-visual software that might aid the autistic and mentally retarded to learn to speak. I was dismayed to find so little available on computer applications. But, then again, the use of computers in special education was virtually nonexistent just two years ago.

This is when the idea for the Young Peoples' LOGO Association came about. None of us are computer scientists, engineers, or programmers. We're a few teachers, media specialists, and journalists that want to see young people enjoy learning about computers and technology.

Initially, we thought we would be corresponding with youngsters in Dallas, New York, and Cambridge -- with youngsters that were already using LOGO in their classes. We were a bit surprised when our initial publicity generated interest from all over the country and from people using all types of computers.

Our second big surprise came when I brought home and Atari 800 system. As a writer, I wanted this system for word processing and for the business aspects of our new organization. But, of course, the games and graphics available on the Atari systems are fascinating. My surprise came in the reaction of the young people. Could LOGO run on the Atari? Could the graphics from the Atari be accomplished in LOGO?

We then began to look at the differences in the two systems. This led to more questions. And it was here that the imagination, the curiosity, and the energy of the young people took over and became infectious.

Young people use the computer for what it is -- a tool. Just as they don't blame the pencil for making a writing mistake, they don't blame the computer for their mistakes. Indeed, they don't view their errors as "mistakes." These are merely bugs to be defined and eliminated. Einstein once said that imagination is more powerful than knowledge. Watching young people at a computer proves his statement.

Our small group of six soon grew to eight and then to twelve. Brothers, sisters, neighbors, and friends of all ages accompanied our software exchange members to the computer club. On one day off from school, Larry called two friends over for an afternoon on the computers. I believe the total that showed up was sixteen.

They wanted to learn about all computers and all languages. When they learned about the club, they wanted to see chapters evolve -- here, at school, around the country, and around the world. We have since acquired a dual disk 64K Apple II Plus, added dual disks and interface to the Atari 800, added Atari 400 systems, and a TRS-80 Color Computer. We have access to TRS-80 Model II and III systems, Osborne, and IBM systems. Letters and phone calls from both young people and adults indicate that our young software exchange group is cartainly not unique. We are fast becoming a lot more than just a users' group.

One of our recently announced programs exemplifies our activities. We invite any young person 18 and under to submit a graphic program to the YPLA software exchange. For the top two programs that make best use of the graphic capabilities of the computer used, we will award a TI99/4 computer and an Atari 400. The programs submitted will be weighted based on the age and experience of the programmer. Other publishers and manufacturers have expressed interest in participating, and I expect that the award list will grow substantially before the end of the contest -- June 30, 1982.

A reporter recently asked why we called this the LOGO association when it was obvious that we were doing so much more. That's true. But that is because the membership has made this so much more.

Let me show you a few of our recent letters. Sense the tone of these -- from young people and adults alike. I could go on with piles of letters from a school for the learning disabled in Pearl Harbor, from a computer club in Clear, Alaska, from an elementary school in North Carolina, from a school in Buenos Aires, from people throughout Canada, from Australia, Saudi Arabia -- even Malta.

We are all very proud to include some noted writers among our members who will be contributing to future issues of our newsletters. The support of publishers and manufacturers has also been very reassuring.

But where does this leave us?

If I may be so bold as to abuse Mr. Lawrence's reference to dynamic activity, the YPLA is nothing more or less than a "dynamic carrot" out in front of the cart designed and built by the membership. We're simply an opportunity for the young and the young-at-heart to explore the exciting and enjoyable world of the microcomputer.

Our approach is simple, straight forward, and very down-to-Earth. It has been said that you can take a horse to water but you can't make him drink. You can teach children, but you can't make them learn. But if you create an environment in which the horse becomes thirsty, he'll drink. And if you create a learning environment where young people become "thirsty" to learn, they will -- because of or maybe in spite of themselves! The YPLA and the world of personal computing is just such an environment.

Take a look at that world for a moment. It is a world of fun, of exciting interactive programs, of dynamic color, sound and graphics. It is also a world of learning -- not only of academics but of decision-making, strategy, logic, confidence, and communication -- for all people, not just the gifted and talented, not just the math lab.

It is a whole new world for the physically and mentally-handicapped, the "dynamic carrot" that can lead them to build a life of self-confidence, self-reliance, and dignity. It is the opportunity for education, communications, productive employment -- the chance to use their creativity and imagination, to contribute to the world around them. A mind -- any mind -- is a horrible thing to waste. What can the computer offer these people? We don't have all of the answers yet, but we're going to find them. Then we're going to tell the rest of the world.

Should you ever doubt the nature of the "carrot," just watch the eyes of a six year old as he sees his procedure run correctly for the first time -- HIS procedure. Watch the concentration of the hyperactive youngster as he pours over the keyboard -- after an hour of programming. We believe this is a most dynamic new world, the potential of which has yet to be fully defined, much less implemented.

The YPLA has a mailing list of about 1,000 now with members in 47 states and six countries. Young people 18 and under can receive our newsletter, Turtle News, at no charge. No one will be excluded for economic reasons. We ask adults to contribute $15 per year to receive Turtle News plus the LOGO Newsletter, an adult-oriented newsletter. Our software exchange disks and tapes are available at $10 each, or they are free for the exchange of a workable program.

We have two active chapters now with others on the way in California, Minnesota, Delaware, Pennsylvania, Florida, Virginia, and Washington. These chapters will be organized like local clubs, following the general pattern of the Scouts. There will be achievement levels, badges, competitions, T-shirts, jackets -- all of the elements to give the young people a sense of belonging to their organization.

The key to this world is LOGO, Pilot, Smalltalk and other new user-friendly computer languages that are themselves "dynamic carrots," enticing young people and the young-at-heart into this expanding world of personal computing. From what our young people have shown us to date, we think it is a pretty nice place to be.

TRAINING TEACHERS FOR THE EFFECTIVE USE OF MICROCOMPUTERS IN SCIENCE EDUCATION

A National Science Foundation Project

Project Directors

Malcolm Wells

and

Gary Bitter
ARIZONA STATE UNIVERSITY
TEMPE, ARIZONA 85287

98

The First Step in Utilizing Computers in Science Education: Preparing Computer Literate Teachers.

In May 1980, the Working Committee on Microcomputers in Education at Arizona State University, initiliated a survey of Arizona Public School Practices and Needs for Computer Assisted Instruction. A survey questionnaire was sent to each school district in Arizona requesting information concerning six computer related issues.[1]

Upon evaluation of these questionnaires, it was found that there was very little use of microcomputers except for courses in computer programming and basic skills in arithmetic. The respondents indicated that approximately one third of the districts used microcomputers to some extent in instructional applications but that 45% of all microcomputers used in the state were being utilized by one district. One conclusion of the committee was that most districts were introducing computers as pilot programs but finding full utilization of this new technology very difficult to attain.

The problems most frequently articulated by the questionnaire respondents were:

1. An acute shortage of personnel who were well trained in the use and implementation of this new technology.
2. A need for assistance in planning computer systems and developing instructional programs.
3. Identification of, and access to effective software.

Among the recommendations presented by the committee were two which could be effectively addressed by way of an academic response. The first of these recommendations was to develop courses dealing with the educational use and implementation of microcomputers. The other recommendation of interest was a recommendation to provide a summer workshop and academic year seminars to assist districts in computer instructional program development.

Our goal for this project was to provide training options and support functions to meet the needs of the school districts for the implementation of educational microcomputer technology.

The successful assimilation of educational computer technology and techniques requires a significant modification of the presently existing educational delivery system. In view of the significant commitment of capital and staff resources necessary to effectively implement programs of this magnitude, it is essential that the district administration and governing board enthuiastically embrace and support the total concept of computer supported education.

In our initial planning phase, we recognized that adequate planning and preparation was imperative to a smooth and effective transition to computer supported education. We considered the following preparatory functions to be essential prerequisites to an orderly transition to this new educational technology.

1. In order to maximize the opportunity for faculty participation in the initial planning process, an in-service.computer awareness program should be conducted for ALL teachers as early as possible.

2. At the conclusion of the in-service awareness program, educational goals and operational objectives should be established to clearly define the mission of the computer related activities in support of the overall educational program.
3. Plan curricular modifications to accomodate any new course offerings as well as alterations to existing course offerings to capitalize upon appropriate computer related activities consistent with the districts goals for computer supported education.
4. Selection of:
   a) Courseware to support the curriculum modifications
   b) Hardware which is compatible with the selected courseware.
5. Plan for student access to computer facilities. Decisions must be made at this time with respect to physical location of the computer hardware as well as the physical configuration of the computer laboratory. Several major considerations follow:
   a) Physical security for computers and support hardware.
   b) Convenience of location for maximum student access.
   c) Availability of supervisory personnel for maximum student access.
   d) Convenience of location for maximum faculty access.

   Possible locations for consideration would be:
   1) The Media Center; media Centers are open and staffed throughout the school day. Media Center personnel

would often times however not be the most competent computer specialists.

2) Self contained computer laboratory; availability of supervisory personnel often times restricts student access.

3) Classroom assignment; classroom assignment provides maximum flexibility but restricts accessibility.

6. Plan and execute in-service training programs to provide specific skills for those teachers who will have responsibility for teaching computer skills under the modified curriculum.

7. Coordination with the counseling staff to insure adequate and accurate dissemination of information to students regarding new program offerings.

8. Coordination with the counseling staff to insure adequate and accurate dissemination of information to parents regarding new course offerings.

9. Planning and preparation for adequate testing procedures to determine the effectiveness.(Determine the extent to which the educational goals are achieved.)

10. Establish a continuing committee to evaluate the computer program on the basis of need for new programs and needed modifications to the existing programs to insure that they support the stated educational goals for computer supported education.

As a direct response to the concerns articulated by the district administrators who responded to the Needs for

Computer Assisted Instruction surveys, a National Science Foundation grant proposal was prepared which would provide computer resource specialists for the educational community within Maricopa County. This proposed training program was to provide training to enable the participants to effectively support their respective districts as they addressed the task of implementing computer wupported educational delivery systems. The project was funded for the summer of 1981 and for the 1981-82 academic year.The details of the funded proposal are as follows:

We recognized that the ultimate success of any program to implement computer supported education requires the unqualified support of the district administration and governing board. Each district was therefore required to submit a letter pledging support for the In-Service training program in their district. In addition, they agreed to grant each teacher who participated in the In-Service programs one hour of credit or professional growth on their respective salary schedules. Active administrative participation in the program was deemed to be an essential aspect of the project.

The project was executed in three phases:

Phase I consisted of a one day general oreintation program for administrators, intended to heighten their level of computer awareness as well as to contribute to their general enthusiasm for computer supported education. Phase I was conducted on the first day of the four week Summer Institute as described in Phase II.

Phase II consisted of a four week Summer Institute at Arizona State University to provide computer training to thirty teachers representing thirteen districts. The participants had a minimum of two years experience in their subject area or grade level. Previous computer experience ranged from none to quite extensive.

The educational goal of this phase of the project was to provide well trained computer resource people within the educational community in Maricopa County. These computer resource people would have skills in:

1. Conducting professional growth workshops in computer literacy.
2. Developing effective software selection criteria.
3. Making hardware selection decisions which are appropriate to implement the computer supported instructional program as developed by each district.
4. Develop and coordinate in-district authoring teams from among interested faculty members.
5. Assist in the development of computer related curriculum.
6. Teach computer related courses.

The academic preparation of the participants, to insure that they were adequately trained to carry out the stated goals, was provided by structured morning lecture activities combined with hands-on computer activities each afternoon. Each participant cooperated in a group authoring project which was presented to the entire group at the conclusion of the Summer Institute. Upon completion of their summer training, these teachers returned to their respective districts to function as computer resource specialists for the educational community within Maricopa County. The skills gained by the participants as a result of this participation will enable them to guide their colleagues in innovative applications of microcomputers to significantly improve the quality of the educational delivery system in their respective districts.

The following topics were presented in the Summer Institute in order to insure that the participants would possess the necessary skills to carry out the above stated objectives:

1. Computer systems and logic -extensive hands-on experience.
2. Introduction to programming techniques.
3. Introduction to computer languages:
    a. machine language
    b. assembly language
    c. BASIC
    d. PASCAL
    e. PILOT
    f. LOGO
4. Authoring techniques.
5. Software selection criteria.
6. Review of software.
7. Hardware selection criteria.
8. Preparation plans for the introduction of computer supported instruction into your district.
9. Computer literacy topics[2]
    a. history of computers

b. computers and society

c. computer related careers

d. what the future holds

10. Preparation of course of study for professional growth workshops.

Phase III of the program was the In-Service aspect in which each participant cooperated in the presentation of an eighteen contact hour computer literacy program to interested teachers from their respective districts. Each presenting team planned their own In-Service program, therefore the program material and structure varied considerably between groups. There were ten teaching teams which ranged from one to eight members who made twelve presentations on ten campuses. This In-Service program reached a total of 272 participants. An example of an In-Service program presented by the Glendale Union, Peroia Unified and Pendergast Elementary group follows. This group met three hours, one afternoooon each week for six weeks.

Requirements:

1. Maintain a notebook or log.

2. Submit at least one summary of an article on instructional use of computers.

3. Presentation of a brief authoring project.

Day 1 - Course Introduction
History of Computers
Types of Computers and Major Components of Computers.

Day 2 - Computers Role in Education
Introduction to Microcomputers

Flowcharting

Day 3 - Courseware Evaluation Criteria

Introduction to BASIC

Day 4 - Courseware Demonstration

Hardware Selection Criteria

BASIC

Day 5 - Courseware Demonstration

Group Work on Authoring Project

Day 6 - Computer Effects on Society/Careers

Courseware Project Presentations

"Where Do We Go From Here"

Final Evaluation

Evaluation of this project has been an on-going process. Throughout the four week Summer Institute, evaluation forms were administered and evaluated daily. The In-Service Institutes have been evaluated by means of an attitudinal survey which was administered on the first day and an evaluation questionnaire which was administered on the last day of each eighteen hour In-Service program. We have periodically modified the program as appropriate and practical in response to participant reaction.

An evaluation of the attitudinal survey indicated that the participants were evenly divided by sex; 55% female. The participants were very experienced teachers, having an average age of 37 years and 12 years of teaching experience. Following are some interesting observations from the attitudinal survey:

1. 89% indicated that they would have participated in the In-Service program even if salary credit had not been

offered.

2. 87% would have welcomed the opportunity to introduce microcomputers into their instructional programs.
3. 64% felt that the implementation of microcomputers would significantly improve the effeciency of the classroom teacher and enhance the sense of job satisfaction.
4. 69% considered the microcomputer to be a permanent part of the educational delivery system.
5. 70% felt that the implementation of microcomputers into instructional applications would not dehumanize the learning process.
6. 72% felt that the introduction of microcomputers into education has provided a rare opportunity to creatively participate in the development of dynamic new educational techniques.
7. 69% felt that microcomputer technology can more effectively provide each student in public education with an individualized instructional program than is presently possible.
8. In-Service participants had a very favorable attitude toward instruction by peers.

The teachers who participated in this program had very high expectations of computers and have very positive attitudes toward computers. It was noted both from interview and from the evaluation of the final questionnaires that the In-Service participants experienced high levels of frustration, resulting from their efforts in BASIC programming.

On the basis of the results of this project, the following are recommendations to be considered in teacher training projects:

1. In-Service participants should spend very little time on authoring or programming techniques. Most of the training should be directed toward user skills.i.e. operating the computer, loading programs from tape and disk, saving programs to tape and disk, operating printers, initializing disks, using educational software.
2. In view of the excellent authoring programs now available, instruction in BASIC programming should be minimized and instruction in the use of a good authoring program should be emphasized.(PILOT for example)
3. Participants should leave the Summer Institute with a structured course of study for the participants in the In-Service programs. The schedule should be outlined for each hour. The In-Service programs should include an examination of courseware programs similar to one each of them might encounter if teaching a course which used Computer Assisted Instruction.
4. More time should be devoted to demonstrating computer class management programs. i.e. Genis I, S.R.A., ZES, gradebook programs, card reader systems.

In conclusion, the project reached more than 300 teachers who provided very positive feed-back on the final evaluation questionnaires. All of our Summer Institute participants are functioning as resource people in their districts and the number of computers used for instructional purposes has incresed significantly in the school districts which supported this program.

We have not yet had an opportunity to initiate another Computer Needs Assessment survey with to the school districts in the state to collect data on the effect of the program on computer supported education. The preliminary data however seems very encouraging.

REFERENCE

1. Survey of Arizona Public School Practices and Needs for Computer Assisted Instruction - May, 1980, G. Bitter, Chair College of Education, Arizonaa State University.

2. Bitter, Gary G. COMPUTERS IN TODAYS WORLD,(Unpublished manuscript) To be published by John Wiley & Sons--January,1982.

THE USE OF THE APPLE COMPUTER WITH YOUNG HANDICAPPED CHILDREN

This microcomputer system was developed by Dr. Laura Meyers at UCLA to create opportunities for control over speech output to enhance general cognitive and language development with non-oral and physically handicapped toddlers and preschoolers. The system is currently used by Dr. Meyers at UCLA's infant intervention program. Dr. Terri Rosegrant will be starting a similar intervention project under Dr. Meyers' direction at Arizona State.

The system includes a 48K Apple Computer with disc drive, a monitor, a printer, a speech synthesizer, a graphics tablet and a touch sensitive keyboard. Software for the system has been developed by Dr. Meyers which allows handicapped children from 18 months to 36 months to use language in a play context.

Dr. Meyers is currently conducting research on the use of this system with handicapped toddlers. Dr. Meyers and Dr. Rosegrant will be extending software development and will be researching its use in early literacy with preschool children.

For further information write to:

Laura F. Meyers
UCLA Intervention Program
UCLA Medical Center
Los Angeles, California 90024

Terri Rosegrant
Early Childhood Education
Arizona State University
Farmer 413 C
Tempe, Arizona 85287

III. ADMINISTRATIVE USES

113

ADMINISTRATORS AND COMPUTERS

NOW AND TOMORROW

James P. Boyle, President
Ombudsman Educational Services, Ltd.
1585 N. Milwaukee Avenue
Libertyville, Il 60048

The tremendous growth of computing power and the dramatic decrease in its cost has been well-documented. Educators everywhere are being forced to adapt to new thinking as to how best to implement and utilize this powerful tool. Unfortunately, education is lagging behind business and industry by about 30 years in computer expertise. Recent years have produced educational conferences, workshops, and in-service programs on the microcomputer. Education is trying hard to play "catch-up" with the technology. As a result a considerable degree of trial and error is occurring and mistakes are being made and are being replicated.

The challenge to the administrators will be to minimize misuse of computers and to develop a sound educational philosophy for immediate and future implementation.

A POWERFUL ADMINISTRATIVE TOOL

The school administrator now has available on his desk, the computing power that was only accessible by renting time on main-frame computers. This time was traditionally used for student and class scheduling and perhaps grade reporting. Student records, attendance, bus routes, personnel records, payroll, inventory, population and income projections are usually performed by hand by all but the largest of school districts.

A rule of thumb for use of microcomputers is that those clerical, arithmetic, storage and retrieval functions that are currently performed by hand can easily be performed more quickly and efficiently by a microcomputer system. To examine just one small area where a microcomputer can be of immediate benefit, the administrator should look at his clerical staff. According to Toffler, the average factory worker is supported by over $25,000 worth of technology. The average secretary is supported by $500 to $1,000 worth of old typewriters. Everyone understands how vital office functions are to any successful operation. Probably no one relates to this more than school administrators who are faced daily with the necessity of speeding the flow of paperwork and information. Office functions, however, are in many cases being performed in almost the same manner they were 100 years ago.

The International Information Word Processing Association in Willow Grove, Pa., estimates that one of 10 office workers now uses some type of word processing equipment. By 1985, it appears that this figure could rise to one of five workers. According to the association, "If it follows the pattern it has, in the 1990's everyone in an office may be at a terminal."

Particularly in times of budget cutbacks and reduced funds, school administrators must look at ways to improve and increase the efficiency of office functions and in so doing keep pace with the technology available to them to accomplish this.

A microcomputer system with a letter quality printer and a $100 word processing program will perform functions of a word processor costing 3 or 4 times as much. Your secretary can now produce work of better quality and in less time. Now, what will the secretary do in time that is saved? The purchase of a data base program for less than $200 will allow her to enter all personnel records on to a disk, mailing labels can be produced, student records can be up-dated, and so can class lists. Still time left over? Purchase Visicalc for budgetting purposes or inventory control or student population projections.

Computer programming is not a criteria for administrative use of the microcomputer. Many administrative programs are already developed and on the market. Student records, student scheduling, attendance, discipline, finances and accounting, long range projections, personnel records, payroll, and inventories are just a few of the programs available to the administrator with a microcomputer at hand. The secretary can now also utilize the microcomputer as a word processor with full screen projection for personalized mailing, address labels, and even have a program to immediately recognize and correct spelling errors.

These are just a few implications for immediate implementation and your first microcomputer purchase evidences money well-spent.

Many companies with access to enormous computer capability from main-frames to minis, are now purchasing microcomputers for their middle and upper executives for their own departments for specialized functions. Using this model a school principal with a budget of $1,000,000 plus should have the availability of personal computing power.

ADMINISTRATIVE DECISIONS

Schools are going to have microcomputers and that is a fact of life. Kids are entering the first grade with knowledge of computer-assisted-instruction, most of them have seen Speak 'N' Spell. Apple will sell twice as many computers in '82 as they did in '81 - nearly 300,000. They are not all being purchased by business, industry and schools. People are buying them for their homes. Homes have kids in them. Parents see what kids do with computers and ask "Why aren't they being used in the schools?" PTA's and other parent groups are buying computers for the schools. Administrators are getting hardware whether they are ready or not. The pressures are mounting on administrators from within (teachers now want computers) as well as from outside influences. As a result, the process of careful selection and deliberation is breaking down. The time to regroup is now. We need to establish priorities in terms of dealing with educational technology.

Making a sound administrative decision about the selection of microcomputers needs to be taken in a very precise and logical order. Too many teachers and administrators, in their haste to join the computer race have acquiesced to the smooth talking hardware salesman. Business and industry has faced this same problem and found themselves with thousands of dollars of hardware only to find suitable programming to be extremely costly additions to insure the hardware will produce what the new owner needs.

In education schools have been led down the path of main-frame, mini-frame and terminals. Microcomputers have been sold on the basis of the largest quantity for the least amount of dollars. Cost effective? Not in the long term!

What happens to the microcomputer in the corner with no software and no one to program its limited capabilities? School administrators must acquire computer literacy and assume a role in the selection of both hardware and software. The process, to be effective, must occur in this order:

1. PHILOSOPHY - how is the computer and CAI going to interface with your school's objectives and philosophical base? Remembering that this is only a tool, and we need to have a basic rationale for its use.

2. PEOPLE - who is going to use this technology? Do we have trained people? Do we have interested people? Will we have a training program? Are people afraid of the computer?

3. SOFTWARE - what programs will fit our philosophy and the needs and interests of the staff? Drill and practice, tutorial, classroom management, literacy, etc. An extensive software review is critical.

4. **HARDWARE** - finally, the actual machine should be selected, not on the basis of low-bid, but rather one that will most clearly meet the adoption criteria listed above. For example, if the administration and staff really does research and development for phases 1, 2, and 3, and the Business Manager buys a machine which cannot satisfy these criteria, the school has bought a tool that will not be utilized effectively.

Clearly the administrator will face many and varied decisions as the computer-wave escalates.

THE FUTURE

And where is all this going to lead us? Schools faced with declining enrollment, loss of Federal aid and reluctant taxpayers at the local level probably wish that the computer would go away, just in terms of dollars. Educators have watched a lot of trends come and go. The "teaching machine" went away as did language labs, the open-classroom and a lot of individualized instruction. Will the computer go away? This one will not go away! We live in an age of rapidly expanding technology in all walks of life.

In Toffler's "The Third Wave", he discusses the Electronic Cottage of the future. In the first wave, when the world was essentially agrarian, most of the essentials of life wre provided from the home food, manufacture, employment, and education. The Industrial Revolution moved society from its home center, agrarian way of life, to the social interaction of urban society. Toffler suggests that the third wave, the Telecommunications Revolution, will return many of life's essentials to the home. Television, home video, video games and personal computers have had a dramatic impact on the outside entertainment industry. The average American worker commutes over 42 miles to work and back. The number of single parent homes has increased significantly. Many workers and employers have learned that through telecommunications many tasks can now be performed at home.

Where is education in light of this trend? If the average secretary is supported by $500 to $1,000 worth of technology, the teacher is supported by $00 worth of technology. We may perhaps have to re-define what needs to be learned and where and how. John Holt advocates "home schooling". The institution, as a place where students come for an education, may be replaced in some degree by students learning at technology. home using telecommunication technology.

The future as always may only be the day after tomorrow. Educational leaders, administrators, need to become as knowledgeable as possible, in a relatively short period of time to deal with "the day after tomorrow". They can be assured that what was true for the last 20 years isn't going to suffice for the next 20.

The bottom line is the necessity for school administrators to lead the way with sound, well thought out guidance in the selection and utilization of the technology of the microcomputer for the benefit of students, staff and administrators.

ADMINISTRATIVE APPLICATIONS...a frame of reference...

Patrick J. Carr

Computer talent for running the schools of tomorrow will be as necessary as the telephone is today. The fact that most educators are not personally or professionally prepared is of little consequence. The words of the poet and songwriter, Bob Dylan, a generation ago seem somehow relevant to the situation confronting us today:

> ...we need to admit that the waters around us have grown;
> and we'd better start swimming or we'll sink like a stone
> for the times they are a'changin...

When I was invited to participate in this conference and asked to speak about administrative applications for microcomputers my reaction, at first, was that there wasn't much one could say about this subject.

In the relatively recent past, microcomputers were used primarily in academic or instructional applications. Administrative needs, usually data processing, were handled by those mainframe monsters or their minicomputer cousins. While the need for using microcomputers was recognized because of lower costs and convenience, the demand simply was not there for administrative applications.

However, if there is one thing we know by now about the microprocessor revolution it is that the relatively recent past soon becomes ancient history. Today, there are a few more administrative software packages but they have yet to withstand the test of time so I cannot recommend any to you this morning.

What I would like to do this morning is to discuss some of the promises and potential for microcomputers in administrative appli-

120

cations; review the state-of-the-art as far as the hardware and software are concerned; and share some purchasing strategies from the brave souls who have gone-on before us, and purchased some microcomputer systems in business, industry, and education.

It is an easy thing to say that the answer to all of our information management needs in education is to purchase computers. No doubt you've already been bombarded by overzealous members of your staff, high pressure sales people and concerned members of the community who somehow got wind of the fact that you are falling behind the times. Computers, they are all proclaiming will solve all of your problems. And, indeed they just might do that -however, you are going to have to work out the details to get them up and running. Let me illustrate this point with a story I heard some time ago:

One morning, President Roosevelt and Will Rogers were standing along the banks of the Potomac. The president expressed concern about a report he had received about enemy submarines having been sighted in the Atlantic. They were headed for Washington, with a mission to sail up the Potomac and attack our nation's capitol. Naturally the president was worried.

Will Rogers put his arm around the president. He assured him that he had little to worry about. "All you need to do," he said, "is heat up the ocean. The enemy will be forced to surface and the Navy will be on stand-by. Boom...bang... that's all she wrote!"

"Indeed," agreed the president, "that would work for sure. But, tell me how do I heat up the ocean?"

"Mr. President, " said Will Rogers, "I've come up with the solution. You are going to have to work out the details!"

Friends, inspite of all the solutions being offered to you for your administrative information management needs; you are going to have to work out the details. The over zealous members of the staff will probably have found another fad; the sales people will be off pushing the latest videoline or fishing tackle; and the concerned members of the community will have found another bone to pick with you. You will be the one left with the computerized system and there are some things that you might want to consider before being pressured into making some hasty decisions.

This morning I would like to share with you a frame of reference that I have developed over the years that might assist you in the decision making process for obtaining a microcomputer system. I hope that you will find this frame of reference helpful in developing a frame of reference of your own.

When discussing computers with educators, I am often reminded of the story about the blind men and the elephant. Remember ?

> It was six men of Indostan
> To learning much inclined
> Who went to see the elephant
> (Though all of them were blind)
> That each by observation might
> satisfy his mind.

One said it was like a wall... Another, like a spear. A rope, a snake, a fan, etc. Educators, it seems to me, often view the computer in much the same way. Talk to those in special ed. or in math/science, reading, music/art, electronics, business, etc. The perspective is from many vantage points, all of which are great but it gets confusing especially when we want to come to terms with the notion of computers in education. I have attempted to work out the concept in a chart and will share it with you in Figure 1.

As you can see, I envision the usage in three general areas and, hopefully, we can somewhat appreciate what other educators are concerned about in making use of computers.

Let us take a look at Section III Administrative Applications.

Any Comments

I would like now to distribute some information that I have received from some of the software vendors. I would suggest that you check them out first before investing unless you have some gambling money in your budget and are in a position to take a chance. You can also spend some time at the conference by visiting the vendors and seeing what they have to offer. They may not have much but at least you will be letting them know that there is an interest and perhaps a market for this product.

In getting into a computer system you need to take care of first things first. While this seems quite obvious it is not always necessarily so...

Do you need an automated information management system? Or can your needs be met with 3-5 index cards?

At one time I worked up the most elaborate management information system for our audio-visual department. It was something else... Unbelievable! It would have provided all sorts of information on vendors, department usage, incidence of repair, etc.- For what? Who was going to use that information? While it might have made an impression on some folks, it would have been a relatively worthless venture, as decisions were not made on the basis of that sort of information. Perhaps, they should be; but in reality they are not. So, what good would my elaborate management information system have been if it could not be useful?

COMMENTS???

Today, I see a lot of computers in our schools. They are buzzing and humming with all sorts of information. Is anyone making use of all that information in some decision making process?

Ok! You are going to automate. The die has been cast; the Rubicon is behind you. First, you need to go shopping for software. That might come to you as a surprise but all of the computers in the world will be of little use if they are not programmed to do what we want them to do. Don't expect the kids or the staff to write programs for you. That is almost like buying typewriters so that someone can produce some literature for the library.

COMMENTS???

Basically, there are three ways to obtain software:

1. Custom designed and written... expensive and time consuming.
2. Packaged or canned programs... great but will they meet your present and future needs. You do not want to get locked into something that will soon be useless.
3. A combination of both of the above ways... be sure that your programs are well documented so that they can easily be revised for future needs.

Programmers??? Good ones are hard to find -they already have jobs. If you find one to moonlight for you he will have to work ingyou in around other demands on his time.

Freelance programmers? Somehow the job they are doing for you isn't as important as the one they are looking for to do for someone else... HUMAN NATURE BEING WHAT IT IS!

Programmers on an hourly basis? A definite No-No! They always seem to need a little bit more time and your budget will in all probability be subject to some restraints.

COMMENTS ???

Let me share with you a few more concerns about getting started. If you go out to get a programmer to meet your needs, do not expect him to help you get your equipment up and running. This is the responsibility of the vendor and the programmer's time will be better spent writing programs.

Also, if you do get a system, be sure that the telephone lines are adequate and that environmental concerns for heat, dust, and electrical current are not ignored. Once a system is set-up do not move it around or you will by buying additional problems.

Comments???

Let me go over briefly the components of a computer system. I have listed them on Figure 2.

Any Comments???

Let us now turn to Figure 3 and review generally some of the basic computer systems available for school administrative applications. These figures are somewhat general and the frame of reference simplified. I hope that it will assist you in coming up with your own reference point in purchasing or determining the type of system that will best meet your needs.

"A ROCK-BOTTOM VIEW OF COMPUTER BASICS"

COMPUTERS ARE TOOLS THAT AID US IN PROCESSING DATA.

DATA IS USED TO PROVIDE INFORMATION THAT CAN BE USEFUL IN MAKING MORE MEANINGFUL DECISIONS.

COMPUTERS FOLLOW A SET OF INSTRUCTIONS CALLED PROGRAMS.

PROGRAMS ARE WRITTEN IN SPECIAL LANGUAGES THAT COMPUTERS UNDERSTAND. ie. FORTRAN, COBOL, BASIC, PASCAL, etc.

THE EQUIPMENT USED TO GET THE SYSTEM RUNNING IS CALLED THE HARDWARE.

THE PROGRAMS ARE GENERALLY REFERRED TO AS THE SOFTWARE.

THE EXTRAS NEEDED TO GET RESULTS FROM THE HARDWARE AND THE SOFTWARE ARE CALLED PERIPHERALS.

HEAVY STUFF? HARDLY! START TO DEVELOP YOUR OWN GLOSSARY TO SERVE AS YOUR FRAME OF REFERENCE AS YOU WORK OUT THE DETAILS FOR YOUR ENTRY INTO THE COMPUTER GAME.

PROMISES AND POTENTIAL OF MICROCOMPUTERS IN EDUCATION

ADEMIC OR INSTRUCTIONAL APPLICATIONS

LEARNING "FROM" "WITH" AND "ABOUT" COMPUTERS

- COMPUTER ASSISTED INSTRUCTION (CAI)
- COMPUTER MANAGED INSTRUCTION (CMI)
- SPECIFIC SUBJECT AREAS:
  - MATH/SCIENCE
  - BUSINESS
  - READING
  - SPECIAL ED.
  - MUSIC/ART
  - ELECTRONICS

- COMPUTER LITERACY
- COMPUTER AWARENESS
- SIMULATION
- INTERACTIVE VIDEO

RESEARCH APPLICATIONS

- ERIC SEARCHES
- DATA ANALYSIS "NUMBER CRUNCHING"
- INFORMATION RETRIVAL FROM LARGER DATA BANKS

FIGURE 1

PROMISES AND POTENTIAL OF MICROCOMPUTERS IN EDUCATION (C'NTD)

NISTRATIVE APPLICATIONS "DATA PROCESSING/MANAGEMENT INFORMATION SYSTEM"

ROLL
ENDANCE
DING
SS SCHEDULING
GET CONTROL
PERTY CONTROL
CTRONIC MAIL
D PROCESSING

EMPLOYEE PERSONAL INFO
HONOR ROLL
SCHOOL CALENDAR
PARKING PERMITS
PRE-REGISTRATION
TUITION RECORDS
STUDENT CLASS LIST

ELECTRONIC BULLETIN BOARD
____
____
____
____

THE LIST IS LIMITED TO YOUR IMAGINATION AND NEED FOR INFORMATION TO MAKE MORE EFFICIENT DECISIONS...

THE BEST SYSTEM IS STILL SUBJECT TO OLD "GIGO"

YOU WILL NEED POSITIVE ATTITUDES AND A CLEARLY DEFINED PURPOSE TO MAKE ANYTHING WORK...

FIGURE 1
(cnt'd)

COMPONENTS OF A COMPUTER SYSTEM

INPUT DEVICES: PUNCHED CARDS, LIGHT PENS, CARD SCANNERS, TYPEWRITER KEYBOARDS, VOICE RECOGNITION DEVICES, ETC.

CENTRAL PROCESSING UNIT (CPU): A MICROPROCESSOR THAT ACTS AS THE HEART OR BRAIN OF THE COMPUTER.

MEMORY: AREA WITHIN THE COMPUTER WHERE INFORMATION IN THE FORM OF INSTRUCTIONS IS STORED.

ROM: READ ONLY MEMORY... PERMANENT INSTRUCTIONS

RAM: RANDOM ACCESS MEMORY... CHANGING INSTRUCTIONS

MEMORY IS EXPRESSED IN TERMS OF K (1,000/1,024 BITS)

OUTPUT FORMAT: SOFT COPY... VIDEO MONITORS, CRT TUBES

HARD COPY: PRINTOUTS

STORAGE: CASSETTE TAPES, FLOPPY DISCS, ETC

FIGURE 2.

COMPUTER SYSTEMS FOR ADMINISTRATIVE APPLICATIONS

	MAINFRAME	MINICOMPUTERS	MICROCOMPUTERS
HISTORY	1946	1950's	1974
COSTS	$250,000+	$10,000+	$99.50 +
SOFTWARE	GREAT	GOOD	POOR
MEMORY	GREAT	GOOD	FAIR-POOR
PROBLEMS	EXPENSIVE INCONVENIENT ???	EXPENSIVE INCONVENIENT ???	AFFORDABLE??? ???
*	*	*	*
*	*	*	*

FIGURE 3

*other important items

Friends, we are all in for some exciting times in the near future in education. Computers will be providing most of the fun. I hope that I have given you some thoughts to plan for the computer revolution so that you will not have to panic.

Any Comments???

Enjoy the rest of the conference...

Patrick J. Carr
Tuba City High School
P.O. Box 67
Tuba City, Arizona 86045

COMPUTER EQUIPMENT ACQUISITION GUIDE
FOR THE EDUCATIONAL ADMINISTRATOR

Tom Keller
Administrative Services Department
Arizona State University
Tempe, Arizona 85287

Acquiring computer equipment for educational purposes can be a confusing and frustrating assignment for an educational administrator. Selecting the appropriate software and hardware components from a growing number of unfamiliar vendors represents a tremendous investment in time to the school official with a limited or non-existent computer background.

There are some procedures, however, that may provide the educational administrator with an organized plan for addressing the many factors inherent in computer equipment selection. The following outline is suggested as one such plan.

EQUIPMENT EVALUATION AND ACQUISITION

I. Identify Participants

II. Develop Instructional Objectives

III. Determine Software Needs

IV. Determine Hardware Needs

V. Develope Bid Specifications and Evaluation Grid

VI. Notify Public of Bid

VII. Hold Bidder's Conference

VIII. Evaluate Bids

IX. Hold a Study Session with Board of Education

X. Make Presentation to Board of Education

I. IDENTIFY PARTICIPANTS

Resource Persons: School Administrator(s)
Teachers (user group)
Advisory Committee (technical experts)
Business Manager/Purchasing Agent
School District Lawyer
Board Member

II. DEVELOP INSTRUCTIONAL OBJECTIVES

Resource Persons: School Administrator
Teachers
Advisory Committee

Procedure: Develop specific objectives for each instructional unit.

Determine the specific role that the computer will play in meeting instructional objectives.

III. DETERMINE SOFTWARE NEEDS

Resource Persons: Teachers
Advisory Committee

Procedure: Research the existing software packages which address the previously defined instructional objectives.

IV. DETERMINE HARDWARE NEEDS

Resource Persons: Teachers
Advisory Committee

Procedure: Identify the hardware by vendor and model that the selected software will run on.

Determine the memory requirements and peripheral devices desired.

V. DEVELOP BID SPECIFICATIONS AND EVALUATION GRID

Resource Persons: School Administrator
Teachers
Advisory Committee
Business Manager/Purchasing Agent
School District Lawyer
Board Member

Procedure: Require that bidders respond to:

- minimum hardware configuration
- individual software packages
- maintanance contract
- user training
- delivery schedule and costs

Develop an evaluation grid to provide a side-by-side evaluation instrument.

VI. NOTIFY PUBLIC OF BID

Resource Persons: School Administrator
Business Manager/Purchasing Agent
Board Member
School District Lawyer

Procedure: Follow district policy for bidding.

VII. HOLD BIDDER'S CONFERENCE

Resource Persons: School Administrator
Teachers
Business Manager/Purchasing Agent
Board Member
Advisory Committee
School District Lawyer

Procedure: Conduct a question and answer session to clarify the bid specifications.

VIII. EVALUATE BIDS

Resource Persons: School Administrator
Teachers
Board Member
Advisory Committee

Procedure: Use the evaluation grid previously developed.

Eliminate all bids which do not meet minimum bid specifications.

Select several "attractive" bids for a hands-on evaluation.

IX. HOLD A STUDY SESSION WITH THE BOARD OF EDUCATION

Resource Persons: School Administrator
Teachers
Business Manager/Purchasing Agent
Advisory Committee
School District Lawyer
Board Member

Procedure: Inform board of procedures followed during the process.

Attempt to clarify technical aspects of the bids.

Provide rationale for your selection as well as rationale for not selecting other bids.

X. MAKE RECOMMENDATION TO BOARD

Resource Persons: School Administrator
Teachers
Business Manager/Purchasing Agent
Advisory Committee

Procedure: Make presentation at an official board meeting recommending your computer configuration.

Microcomputers in the Practice of School Psychology

Marley W. Watkins
Deer Valley School District
17624 N. 31st Avenue
Phoenix, Arizona 85007

School psychologists provide a diversity of services to the public and private schools of America. Typically, such activities as testing, counseling, consultation, program development, research and in-service training are considered to be within their purview (Senff & Snider, 1980; Holt & Kicklighter, 1972). However, surveys of school psychologists' work schedules reveal that a major portion of their time is devoted to psychoeducational evaluation activities (Cook & Paterson, 1977; Fairchild, 1974; Keogh et al., 1975). This indicates that the 'average' school psychologist's daily work is composed primarily of mandated activities (i.e., testing and evaluation) while non-mandated services (i.e., consultation, research, etc.) languish. This situation can be expected to continue unabated or to intensify unless the individual psychologist modifies his or her behaviors and thereby alters the balance of mandated to non-mandated activity. Many school psychologists have investigated computer technology for its potential value as a tool to assist them in the modification of their roles, and have determined that the technology offers considerable promise. One major advantage lies in utilizing computer technology to increase the efficiency with which mandated activities are performed so as to allow a diversion of time and energy to non-mandated functions. This is 'working smarter rather than working harder.'

Computerized Administration, Scoring and Interpretation of Tests

Applications of computer technology in psychology and psychiatry span several decades. Altemose and Williamson (1981) reported that systems for scoring and interpreting the MMPI were developed in the early 1960's. More recently, a computerized administration of the MMPI was found to be equivalent to the printed version (Lushene, O'Neil & Dunn, 1974). Kleinmuntz and McLean (1968) demonstrated that a computer could administer MMPI items in an "intelligent" manner and that the average clinician was "outperformed" by a computerized system for interpretation of the MMPI (Kleinmuntz, 1975). Thus, the computer can assist the psychologist by administering, scoring, or interpreting existing educational and psychological tests. It has been employed in this manner to self-report personality measures (Altemose & Williamson, 1981), spelling (Hasselbring & Crossland, 1981), parental reports of child behaviors (Greene et al., 1981) and intelligence (Elwood, 1972). By allowing the computer to perform clerical tasks associated with the testing process, the psychologist can devote an equivalent amount of time to alternative activities. Of course, caution must be exercised. The reliability and validity of automated versions of conventional paper-and-pencil tests have not been investigated to any satisfactory extent (Hansen, 1971). These two attributes must be of paramount importance if results are to be used to make decisions about children (American Psychological Association, 1974). Computerized scoring is relatively straight forward, but mechanized interpretation is subtle in its complications. Interpretations of psychological tests can be based upon sound clinical and statistical principles or upon a concretized, overly-simplistic clinical approach in isolation. Users must insure that computerized interpretions which affect children are not only impressive in appearance but are, in addition, reliable and valid in operation (American Psychological Association, 1972).

Tailored Testing

The psychologist may also utilize the computer for tests which were tailored for administration, scoring and interpretation on the computer itself. Rather than providing the same set of items for all examinees, as traditional paper-and-pencil tests do, these tailored tests allow for items to be selected as the process of testing proceeds, with items presented based upon previous responses (McKinley & Reckase, 1980). In many ways, these tests are tailored to fit the student so that a more efficient testing process ensues. At present, this application of computer technology has been implemented primarily on large computers, but a mathematics error analysis program for a microcomputer has recently been introduced (Watkins, 1982). Tailored testing will become more prevalent in the near future as microcomputers increase in power and as the attributes of this testing mode are understood in greater detail.

Computer Assisted Diagnosis

A most powerful application of computer technology to school psychology practice resides in computer assisted diagnosis. The end result of the psychologist's testing activities is a diagnostic classification designed to "identify the problem .... in such a way that an appropriate and restorative treatment may be carried out" (Gough, 1971, p. 160). That is, provision of effective remedial and preventative services is dependent upon accurate identification of the specific nature and course of the child's problem. Unfortunately, there is considerable evidence to question the reliability and validity of many diagnostic formulations (Algozzine & Ysseldyke, 1981; Spitzer & Fleiss, 1974; McDermott, 1980a; Johnson, 1980; Achenbach & Edelbrock, 1978; Cantwell, Russell, Mattison & Will, 1979; Freeman, 1971).

The nature and remediation of erroneous psychodiagnostic observations is intimately intertwined with the utility of computer assisted diagnosis. The computer may aid the psychologist in overcoming some common specific psychodiagnostic errors and may remove a significant portion of the substantial information processing demands which in themselves generate diagnostic errors (Lueger & Petzel, 1979). In the later case, Chapman and Chapman (1967) have noted that "the clinician's task often apparently exceeds the capacity of the human intellect .... the ultimate solution ... may lie in at least a partial replacement of clinical psychodiagnostic methods by actuarial prediction" (p. 204). Specific sources of psychodiagnostic error have been identified by Kahneman and Tversky(1973,1982), Crocker (1981), Snyder, Shenkel and Lowery (1977), Arkes (1981), and McDermott (1981). Covariation misestimation, preconceived expectancies, lack of awareness of one's own judgment process, overconfidence and hindsight bias (Arkes, 1981) have been demonstrably evident in human decison making but can be mitigated through computer assisted diagnosis.

Efforts to overcome diagnostic instability by applying computer technology have been made in medicine and psychiatry as well as in psychology. The Internist program is a computerized diagnostic program which covers over eighty percent of the diagnoses of internal medicine and MYCIN advises physicians in the appropriate treatment of infections (Szolovits & Pauker, 1978). DIAGNO III (Spitzer, et al., 1974) relies upon structured psychiatric interviews to arrive at as many as 75 of the discrete psychiatric diagnoses listed in the American Psychiatric Association's diagnostic manual. However, the computer can do no better than the data it is given (Spitzer & Endicott, 1974). DIAGNO is based upon interview data and classifies into DSM categories. Significant constraints on its diagnostic validity reside in the instability of interview data and in the ambiguity found in DSM categorizations. Nevertheless, computerized systems have been found to be at least as reliable as expert clinicians in the diagnosis of adult clinical disorders (Johnson, Klinger, Giannetti & Williams, 1980).

A computer assisted diagnostic system developed by McDermott (1980b) exhibits remarkable promise for the reliable differential diagnosis of problems in children. This multidimensional actuarial classification (MAC) system incorporates the logical sequencing and efficiency of modern diagnostic decision theory (Dailey, 1971) and systems analysis(Nathan, 1967), as well as the objectivity and reliability of Quay's (1972) quantitative model. MAC consists of a series of routines that apply statistical rules in hierarchical order to standardized psychometric, observational, and interview data to result in such classifications as mental retardation, learning problems, social and behavioral maladjustment, etc. The current version of MAC accepts diagnostic data from: intelligence tests such as the WISC-R and WAIS, academic achievement measures such as the Wide Range Achievement Test, adaptive behavior indices such as the Vineland Social Maturity Scale, and behavioral observations such as those from the Bristol Social Adjustment Guides.

Each of MAC's decision rules is modeled after nationally accepted standards for differential diagnosis. For example, the American Association on Mental Deficiency (1977) criteria are employed for defining mental retardation. The definition of learning disability was based upon the concepts elucidated by Thorndike (1963) and McLeod (1979), and are fully compatible with the standards specified by the Education for All Handicapped Children Act (U. S. Department of HEW, 1977). MAC applies appropriate distributional, relational, reliability, and validity statistics to conduct series of actuarial tests and, within the framework of professionally validated, sequenced decision rules, renders diagnostic statements. Thus, a complex combination of statistical and logical decision tree reasoning (Fleiss, et al., 1972) is utilized by MAC to produce maximally consistent, statistically sound diagnostic categorizations. Such a melding of categorical and probabilistic reasoning was described by Szolovits and Pauker (1978) as offering the greatest promise for producing a program which could demonstrate expertise in medical consultation. Likewise, a union of clinical and statistical methodologies has been recommended for advancement of psychodiagnostic judgments (Pankoff & Roberts, 1968; Einhorn, Kleinmuntz & Kleinmuntz, 1979). MAC has accomplished the consolidation of approaches endorsed by many experts and has embodied components which one author suggested would require a computer program developed at a national level (Page, 1980).

The promise of multidimensional actuarial classification is now being realized. Diagnostic classifications of child psychopathology generated by MAC were tested against the criterion of expert psychological opinion and yielded agreement at a level 86 percent beyond chance (McDermott & Hale, in press). Hale and McDermott (in press) investigated the construct and predictive validity of MAC by using it to group 200 school children who had been referred for psychological services. Factor analysis of the resultant grouping patterns revealed seven classifications which were found to be congruent with major categories currently accepted in clinical and quantitative schools of child psychopathology.

Computer assisted diagnosis in school psychology is a new, exciting application of computer technology. Its very newness mandates that additional validation research be conducted, but the MAC system has demonstrated that it can in this stage of its development serve the school psychologist as an efficient diagnostic aid. Although currently available only for mainframe computers, an expanded version of MAC is currently being prepared for operation on a microcomputer (McDermott & Watkins, in press). With this tool, the school psychologist can increase the efficiency and effectiveness of the psychodiagnostic process.

Report Writing

A major, and currently available, use of microcomputers in school psychology is word processing. Almost all testing conducted by the psychologist results in an evaluation report. Production of such reports is often a time consuming clerical task which detracts from other, more valuable pursuits. Many commercially available word processing programs will convert the psychologist's microcomputer into an 'electronic typewriter.' Manipulation of textual materials is facilitated by these programs and they can be used to create highly individualized, yet rapid evaluation reports suitable for immediate use in multidisciplinary conferences. The storage, retrieval and manipulation of text made possible by these programs allows the psychologist to establish a data base of commonly used phrases, sentences, paragraphs, objectives, etc. and thereby considerably reduces the time required for report writing. Current software technology even makes it possible for the computer to check the accuracy of the psychologist's spelling! The end result is a typed, justified, professional document in less time.

Research and Program Evaluation

The microcomputer can also serve as a valuable adjunct to the psychologist in areas other than testing and diagnosis. The microcomputer is, for example, consummately suited for research and program evaluation. Statistical analyses required for research and evaluation may be performed on the computer and data files can be maintained, manipulated, searched and retrieved. In short order, capacities of the microcomputer will rival the mainframe computers used by many psychologists during their graduate statistics and measurement courses. These capabilities offer the psychologist an exciting opportunity to expand his or her role to include research and program evaluation functions.

Consultation

Computer assisted instruction is expanding into the nation's schools and has made inroads in all disciplines, including special education (Watkins & Webb, 1981). The school psychologist who is familiar with the concepts and operations of this new educational vehicle will be a valuable resource to any school system. Consultation opportunities, both case centered and system wide, will invariably be expanded for the psychologist with these competencies. Exploiting microcomputers to enrich the curriculum of gifted students, to serve as reinforcement in behavioral managment programs, to assist with preparation of IEP's, to maintain attendance records, to increase the socialization contacts between special and regular education students, to improve home-school communications and to analyze textbook readibility are but a few innovation applications with which the school psychologist might become involved. In many cases no other school staff member will possess all the knowledge and competencies required for fruitful implementation of microcomputer technology. The school psychologist who is prepared and industrious can in this way move teacher perceptions away from the exclusively testing role toward acceptance as a consultant in education and psychology capable of competent assistance in curriculum development, in-service training, etc.

Counseling

Utilizing the microcomputer as an aid in counseling and therapy is an intriguing application which has, as yet, received little attention. Dickson and Wanderer (1981) reported a case where the microcomputer was used as a therapeutic tool by the psychologist with "an angry, aggressive, deliquent youth (who) changed dramatically in less than six weeks ..." (p. 129). This area is completely unexplored but is worthy of investigation. Many children are fascinated by computers (doubters should observe at any electronic game arcade) and are therefore more communicative with the psychologist who shares a like interest than might otherwise be the case. This seems to be particularily true with hostile, unmotivated children. The psychologist who skillfully employs the microcomputer in therapy may be offered a small edge in breaching the child-adult chasm that so often obstructs effective communication with these children.

Conclusion

Application of computer technology to the practice of school psychology was described by Altemose and Williamson (1981) within the framework of a discussion on the threat posed to the job security of school psychologists. They introduced two models of computer usage: the intelligent computer and the computer-as-assistant model. In the former, the computer would think, reason, and learn but in the later would only serve as a tool. Such a distinction is spurious. The computer is employed as a tool in business (Felsen, 1978), medicine (Szolovits & Pauker, 1978), neurophysiology (Bradley, 1982), geology (Gunther, 1982), etc. and will also be utilized in that capacity in the practice of school psychology. It is merely another example of humans inventing a tool to augment their physicial and mental power. As the bulldozer carves mountainsides, the computer channels unruly rivers of information. Both tools accomplish their massive tasks at the behest of Homo sapiens. Clearly, what we have is the tool of a new order, a quasi-intelligent assistant.

The integration of computer technology into the practice of school psychology must be engineered by school psychologists to insure that the richest possible amalgamation precipitates to the school children of America. It is hoped that the preceding discussion will focus and guide this crucial process.

References

Achenbach, T. M., & Edelbrock, C. S. The classification of child psychopathology: A review and analysis of empirical efforts. Psychological Bulletin, 1978, 85, 1275-1301.

Algozzine, B., & Ysseldyke, J. E. Special education services for normal children: Better safe than sorry? Exceptional Children, 1981, 48, 238-243.

Altemose, J. R., & Williamson, K. B. Clinical judgment vs. the computer: Can the school psychologist be replaced by a machine? Psychology in the Schools, 1981, 18, 356-363.

American Association on Mental Deficiency. Manual on terminology and classification in mental retardation. Washington, D.C.: Author, 1977.

American Psychological Association. Ethical standards of psychologists. Washington, D. C.: Author, 1972.

American Psychological Association. Standards for educational & psychological tests. Washington, D. C.: Author, 1974.

Arkes, H. R. Impediments to accurate clinical judgment and possible ways to minimize their impact. Journal of Consulting and Clinical Psychology, 1981, 49, 323-330.

Bradley, R. M. Tapping into the brain. Microcomputing, 1982, 6, 72-79.

Cantwell, D. P., Russell, A. T., Mattison, R., & Will, L. A comparison of DSM-II and DSM-III in the diagnosis of childhood psychiatric disorders. Archives of General Psychiatry, 1979, 36, 1208-1213.

Chapman, L. J., & Chapman, J. P. Genesis of popular but erroneous psychodiagnostic observations. Journal of Abnormal Psychology, 1967, 72, 193-204.

Cook, V. J., & Paterson, J. G. Psychologists in the schools of Nebraska: Professional functions. Psychology in the Schools, 1977, 14, 371-376.

Crocker, J. Judgment of covariation by social perceivers. Psychological Bulletin, 1981, 90, 272-292.

Dailey, C. A. Assessment of lives. San Francisco: Jossey-Bass, 1971.

Dickson, K., & Wanderer, Z. Psychology and the Apple: Pioneer use of an Apple in psychotherapy opens new vistas for treatment. Softalk, 1981, 2, 125-130.

Einhorn, H. J., Kleinmuntz, D. N., & Kleinmuntz, B. Linear regression and process-tracing models of judgment. Psychological Review, 1979, 86, 465-485.

Elwood, D. L. Automated versus face-to-face intelligence testing: Comparison of test-retest reliabilities. International Journal of Man-Machine Studies, 1972, 4, 363-369.

Fairchild, T. A. An analysis of the services performed by a school psychologist in an urban area. Psychology in the Schools, 1974, 11, 275-281.

Felsen, J. Low-cost, personal computer-based investment decision systems. New York: CDS Publishing Company, 1978.

Fleiss, J. L., Spitzer, R. L., Cohen, J., & Endicott, J. Three computer diagnosis methods compared. Archives of General Psychiatry, 1972, 27, 643-649.

Freeman, M. A reliability study of psychiatric diagnosis in childhood and adolescence. Journal of Child Psychology and Psychiatry, 1971, 12, 43-54.

Gough, H. Some reflections on the meaning of psychodiagnosis. American Psychologist, 1971, 26, 160-167.

Greene, R. L., Martin, P. W., Bennett, S. R., & Shaw, J. A. A computerized scoring system for the Personality Inventory for Children. Educational and Psychological Measurement, 1981, 41, 233-235.

Gunther, F. J. Uncovering the history of the earth. Microcomputing, 1982, 6, 72-79.

Hale, R. L., & McDermott, P. A. Pattern analysis of an actuarial strategy for computerized diagnosis of childhood exceptionality. Journal of Learning Disabilities, in press.

Hansen, D. N. Review of automated testing. Florida State University Computer-Assisted Instruction Center, 1971. (ERIC Document Reproduction Service No. ED 050 553)

Hasselbring, T. S., & Crossland, C. L. Using microcomputers for diagnosing spelling problems in learning-handicapped children. Educational Technology, 1981, 21, 37-39.

Holt, F. D., & Kicklighter, R. H. (Eds.). Psychological services in the schools: Readings in preparation, organization and practice. Dubuque, Iowa: Wm. C. Brown, 1972.

Johnson, J. H., Klinger, D. E., Giannetti, R. A., & Williams, T. A. The reliability of diagnosis by technician, algorithm, and computer. Journal of Clinical Psychology, 1980, 36, 447-451.

Johnson, V. M. Analysis of factors influencing special education placement decisions. Journal of School Psychology, 1980, 18, 191-202.

Kahneman, D., & Tversky, A. On the psychology of prediction. Psychological Review, 1973, 80, 237-251.

Kahneman, D., & Tversky, A. The psychology of preferences. Scientific American, 1982, 246, 160-173.

Keogh, B. K., Kukic, S. J., Becker, L. D., McLaughlin, R. J., & Kukic, M. B. School psychologists' services in special education programs. Journal of School Psychology, 1975, 13, 142-147.

Kleinmuntz, B. The computer as clinician. American Psychologist, 1975, 30, 379-387.

Kleinmuntz, B., & McLean, R. S. Diagnostic interviewing by digital computer. Behavioral Science, 1968, 13, 75-80.

Lueger, R. J., & Petzel, T. P. Illusory correlation in clinical judgment: Effects of amount of information to be processed. Journal of Consulting and Clinical Psychology, 1979, 47, 1120-1121.

Lushene, R. E., O'Neil, H. F., & Dunn, T. Equivalent validity of a completely computerized MMPI. Journal of Personality Assessment, 1974, 38, 353-361.

McDermott, P. A. Congruence and typology of diagnoses in school psychology: An empirical study. Psychology in the Schools, 1980a, 17, 12-24.

McDermott, P. A. A systems-actuarial method for the differential diagnosis of handicapped children. Journal of Special Education, 1980b, 14, 7-22.

McDermott, P. A. Sources of error in the psychoeducational diagnosis of children. Journal of School Psychology, 1981, 19, 31-44.

McDermott, P. A., & Hale, R. L. Validation of a systems-actuarial computer process for multidimensional classification of child psychopathology. Journal of Clinical Psychology, in press.

McDermott, P. A., & Watkins, M. W. Micro MAC. In preparation.

McKinley, R. L., & Reckase, M. D. Computer applications to ability testing. AEDS Journal, 1980, 13, 193-203.

McLeod, J. Educational underachievement: Toward a defensible psychometric definition. Journal of Learning Disabilities, 1979, 12, 42-50.

Nathan, P. E. Cues, decisions, and diagnoses: A systems-analytic approach to the diagnosis of psychopathology. New York: Academic Press, 1967.

Page, E. B. Tests and decisions for the handicapped: A guide to evaluation under the new laws. Journal of Special Education, 1980, 14, 423-483.

Pankoff, L. D., & Roberts, H. V. Bayesian synthesis of clinical and statistical prediction. Psychological Bulletin, 1968, 70, 762-773.

Quay, H. C. Patterns of aggression, withdrawal and immaturity. In H. C. Quay & J. S. Werry (Eds.). Psychopathological disorders of childhood. New York: Wiley, 1972.

Senft, L. B., & Snider, B. Elementary school principals assess services of school psychologists nationwide. Journal of School Psychology, 1980, 18, 276-282.

Snyder, C. R., Shenkel, R. J., & Lowery, C. R. Acceptance of personality interpretations: The "Barnum Effect" and beyond. Journal of Consulting and Clinical Psychology, 1977, 45, 104-114.

Spitzer, R. L., & Endicott, J. Can the computer assist clinicians in psychiatric diagnosis? American Journal of Psychiatry, 1974, 131, 523-530.

Spitzer, R. L., Endicott, J., Cohen, J., & Fleiss, J. L. Constraints on the validity of computer diagnosis. Archives of General Psychiatry, 1974, 31, 197-203.

Spitzer, R. L., & Fleiss, J. L. A reanalysis of the reliability of psychiatric diagnosis. British Journal of Psychiatry, 1974, 125, 341-347.

Szolovits, P., & Pauker, S. G. Categorical and probabilistic reasoning in medical diagnosis. Artificial Intelligence, 1978, 11, 115-144.

Thorndike, R. I. The concepts of over- and underachievement. New York: Teachers College, 1963.

U. S. Department of Health, Education and Welfare. Assistance to states for education of handicapped children. Federal Register, 1977, 42, 65082-65085.

Watkins, M. W. Diagnostic mathematics test: Addition. SouthWest EdPsych Services: Phoenix, in preparation.

Watkins, M. W., & Webb, C. J. Computer assisted instruction with learning disabled students. Educational Computer Magazine, 1981, 1, 24-27.

MICROCOMPUTERS - TOOLS IN THE CLASSROOM
- TUTORS AT HOME

Presented by Pat Kuhn
Phoenix Elem. School District
9068 N. 28th Street
Phoenix, Arizona 85028

WHAT IS THE MICROCOMPUTER?

1. Computers/Microcomputer
2. Performance and Breakthroughts
3. Parts of a Computer System
4. Capabilities and Shortcomings

WHAT THE MICROCOMPUTER OFFERS TO EDUCATION?

1. Opportunity for Great Teaching
2. Increased Student Motivation
3. Immediate Feedback
4. Individualized Instruction
5. Case for Microcomputers
6. Responsibilities

HOW TO GET STARTED FOR CLASSROOM/CENTER USE

1. What Configuaration Do You Need?
2. Where to Put it
3. One' Class - One Computer
4. Portable Computer
5. Computer Center
6. Student Reaction: How to Exploit It?
7. Teacher Reaction: How to Overcome It?

MICROCOMPUTER USES IN INSTRUCTION

1. Computer Science
2. Problem - Solving
3. Instructional Simulations
4. Games
5. Computer Assisted Instruction (CAI)
    a. Drill & Practice
    b. Tutorial
6. Administrative Management
    a. Student Information
    b. Test Scores/Analysis
    c. Student Prescriptions

MICROCOMPUTER USES IN THE HOME

1. Personal Uses:
    a. Instruction
    b. Communications
    c. Recreational
2. Home Control
    a. Finance Programs
    b. Household/Appliance Operations
    c. Security

WHERE TO GO FOR HELP?

1. Special Interest Groups - Publications
2. User Groups
3. Clearing houses
4. State Organizations
5. Conferences

DEMONSTRATION OF EDUCATIONAL PROGRAMS ON THE PET COMPUTER

SLIDE PRESENTATION - Students and Computers

HANDOUTS
- Outline of Presentation
- Copies of Instruction
    - Programs -
- Software Resource Guide

LOWEST TERM FRACTIONS

use PLACE VALUE tape

*"FRACTIONS IN THEIR LOWEST TERMS"(TITLE - LINE 92)

*"1/3 2/3 7/8 1/6" (REPEATED INFO. - TOP OF SCREEN)
"2/5 1/4 5/6 3/5" (LINES 95 - 100)
"3/4 1/2 1/5 3/8"

* "4/10 6/15 8/20", 2/5 (DATA - LINES 600 - 700)
"14/16 21/24 28/32", 7/8
"2/4 3/6 4/8", 1/2
"10/12 15/18 20/24", 5/6
"2/6 3/9 4/12", 1/3
"6/10 9/15 12/20", 3/5
"6/8 9/12 12/16", 3/4
"2/12 3/18 4/24", 1/6
"2/10 3/15 4/20", 1/5
"6/16 9/24 12/32", 3/8
"2/8 3/12 4/16", 1/4
"4/6 6/9 8/12", 2/3

*"READ THE EQUIVALENT FRACTIONS." (DIRECTIONS - LINES 15 - 22)
"CHOOSE THE LOWEST TERM FRACTION"
"THAT IS EQUAL TO THEM. TYPE THE LETTER"
"OF THAT FRACTION, THEN PRESS RETURN."

* Q$; " ARE EQUAL TO " ;A$;"." (CORRECT ANSWER - LINE 190)

LIQUID MEASUREMENT

use PLACE VALUE tape

* "LIQUID MEASUREMENT"   (TITLE - LINE 92)

* "TABLESPOON   PINT"   (REPEATED INFO. - TOP OF SCREEN)
"QUART   CUP"   (LINES 95 - 100)
"GALLON   HALF GALLON"

* "2 PINTS", QUART   (DATA - LINES 600 - 700)
"16 TABLESPOONS", CUP
"4 QUARTS", GALLON
"3 TEASPOONS", TABLESPOON
"2 CUPS", PINT
"2 QUARTS", HALF GALLON
"4 CUPS", QUART
"4 PINTS", HALF GALLON
"8 CUPS", HALF GALLON
"2 HALF GALLONS", GALLON
"32 TABLESPOONS", PINT
"8 PINTS", GALLON

* "DECIDE WHICH MEASUREMENT IS EQUAL"   (DIRECTIONS - LINES 15 - 22)
"TO THE AMOUNT GIVEN. TYPE YOUR"
"ANSWER AND PRESS RETURN."

* Q$; " ARE EQUAL TO " ;A$;"." (CORRECT ANSWER - LINE 190)

TIME

use PLACE VALUE tape

```
*"TIME"   (TITLE - LINE 92)

* " A 9:45      D  8:00      G  10:10"     (REPEATED INFO. - TOP OF SCREEN)
  " B 2:15      E  3:45      H   4:30"              (LINES 95 - 100)
  " C 6:30      F 11:35      I   1:55"

* "HALF PASSED FOUR", H          (DATA - LINES 600 - 700)
  "QUARTER TO TEN", A
  "EIGHT O'CLOCK", D
  "TWENTY-FIVE TO TWELVE", F
  "TEN AFTER TEN", G
  "QUARTER AFTER TWO", B
  "FIFTEEN MINUTES TO FOUR", E
  "SIX THIRTY", C
  "TEN AFTER TEN", G
  "FIVE TO TWO", I
  "FOUR THIRTY", H
  "FIFTEEN MINUTES TO TEN, A

* "READ THE PHRASE.  DECIDE WHICH"
  "TIME LISTED ABOVE IS THE SAME TIME"
  "AS THE PHRASE YOU READ.  TYPE ITS"
  "LETTER AND PRESS RETURN."

* Q$; " IS WRITTEN AS " ;A$; "."    (CORRECT ANSWER - LINE 190)
```

PLACE VALUE

```
1 PRINT"⊐
2 REM **** Q$=QUESTION,A$=ANSWER
3 REM **** N$ = STUDENT NAME
4 PRINT"▩▩▩▩▩▩▩▩
5 PRINT"TYPE YOUR FIRST NAME AND PRESS THE      RETURN BUTTON."
6 PRINT"▩▩▩
7 INPUT N$
8 PRINT"▩▩▩
9 PRINT"OK, " N$"."" LET'S GET STARTED!"
10 FOR K=1TO8000:NEXT
11 PRINT"⊐▩▩▩▩▩▩▩▩▩
12 PRINT"____________________________________________"
14 PRINT
15 PRINT"READ THE NUMBER.  DECIDE ON THE"
16 PRI[illegible]
17 PRINT"PLACE VALUE OF THE NUMBER IN THE ▨BOX■."
18 PRINT
19 PRINT"TYPE THE LETTER OF THE CORRECT ANSWER"
20 PRINT
21 PRINT"ANS PRESS RETURN."
22 PRINT
23 PRINT"____________________________________________"
40 W=0:C=0:V=0
49 FORK=1TO7000:NEXT
65 PRINT"⊐
66 PRINT"▩▩▩
90 A=RND(2). : GOSUB 91: GOTO 105
91 PRINT"⊐
92 PRINTTAB(12)"▨ PLACE VALUE "
94 PRINT"____________________________________________"
95 PRINT"▨A■ HUNDREDS  ▨D■ TENS           ▨G■ ONES"
96 PRINT"____________________________________________"
97 PRINT"▨B■ HUNDRED   ▨E■ TEN THOUSANDS ▨H■ THOUSANDS   THOUSANDS"
99 PRINT"____________________________________________"
100 PRINT"▨C■ HUNDRED   ▨F■ TEN MILLIONS  ▨I■ MILLIONS    MILLIONS"
102 PRINT"____________________________________________"
103 PRINT"▩▩
104 FOR K=1TO3000:NEXT : RETURN
105 V=V+1
110 READ Q$,A$
115 IF A$="99" THEN 895
120 PRINT Q$;
123 PRINT
125 PRINT
130 INPUT R$
135 PRINT
140 IF R$ =A$ THEN 302
145 PRINT
150 PRINT " NO, TRY AGAIN."
153 FORK=1TO1000:NEXT
155 PRINT
160 INPUT R$
165 PRINT
170 IF R$ =A$ THEN 302
175 PRINT
181 M=INT(4*RND(1)+1)
182 PRINT"⊐
183 IF M=1 THEN PRINT "NO, THAT'S NOT IT."
```

```
184 IF M=2 THEN PRINT "NO, KEEP TRYING!"
185 IF M=3 THEN PRINT "DON'T GIVE UP."
186 IF M=4 THEN PRINT "WRONG, TRY AGAIN."
187 FOR K=1TO3000:NEXT
188 GOSUB 91: PRINT "QQ"
189 PRINT "     THE CORRECT HNSWER IS:   ";A$;
191 PRINT"QQQQ
192 FORK=1TO9000:NEXT
195 W=W+1
200 IF W=6 THEN 330
210 GOTO 90
300 REM **** SUB TO HANDLE CORRECT ANSWER"
302 M=INT(4*RND(1)+1)
304 IF M=1 THEN PRINT"VERY GOOD!"
305 FORK=1TO1000:NEXT
306 IF M=2 THEN PRINT "EXCELLENT!"
307 FOR K=1TO1000:NEXT
308 IF M=3 THEN PRINT "KEEP IT UP!"
309 IF M=4 THEN PRINT "NICE WORK!"
310 FOR K=1TO2000:NEXT
313 PRINT
314 C=C+1
315 IF V=12 THEN 440
320 GOTO 90
330 PRINT
335 PRINT"]
336 PRINT"QQQQQQ
340 PRINT"]
350 PRINT"YOU SCORED" C;"CORRECT"
352 PRINT
354 PRINT"AND" W;"INCORRECT OUT OF"
356 PRINT
358 PRINT V;"VOCABULARY WORDS YOU ANSWERED."
359 PRINT
400 PRINT "LET'S TRY AGAIN!"
402 PRINT
405 PRINT"________________________________________"
410 GOTO 510
430 PRINT
440 PRINT"______________________________________"
450 PRINT"YOU SCORED" C;"CORRECT"
455 PRINT
460 PRINT"AND" W;"INCORRECT OUT OF"
462 PRINT
465 PRINTV; "VOCABULARY WORDS YOU ANSWERED."
470 PRINT
480 PRINT
500 PRINT "YOU DID A GREAT JOB, KEEP IT UP!"
501 PRINT
503 PRINT"________________________________"
504 PRINT
505 PRINT" DID YOU HAVE FUN?
506 PRINT
505 PRINT
510 PRINT "DO YOU WANT TO PLAY AGAIN?"
511 PRINT
512 PRINT "TYPE 1=YES OR 2= NO  AND PRESS RETURN"
513 INPUT P
514 PRINT
515 IF P=1 GOTO10
```

```
 516 PRINT
 5 GOTO P OR1 GOTO 10
 520 END
 600 DATA "23,36■47",A
 605 DATA "133■,042,926",I
 610 DATA "237■,053", RH
 615 DATA "36■43,258,901",C
 620 DATA "206,35■9",D
 630 DATA "5,37■40,912",B
 640 DATA "43,869,534■",G
 650 DATA "629■5,475",E
 655 DATA "537■6,410,599",F
 660 DATA "234■3,675",E
 670 DATA "7,973■.206",H
 680 DATA "934■5,308,178",F
 690 DATA"195,831■2",D
 700 DATA ""7,426,6,390",B
 895 RESTORE
 900 GOTO 100
 999 END
 6000 DATA "23,36■47",
READY.
```

SILLY STORIES

```
1 PRINT"⊐
2 PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━"
4 FORK=1TO10
5 PRINT"▮▮            SILLY STORIES            ▮▮"
6 NEXT
7 PRINT"━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
8 REM **** BY PAT KUHN, PHOENIX, ARIZONA
9 FOR K=1TO 9000:NEXT
10 PRINT" ⊐▩▩▩▩▩
11 PRINT"WHAT IS YOUR FIRST NAME?"
12 PRINT"▩▩▩▩▩
20 INPUT N$(1)
21 PRINT"▩▩▩▩▩
22 PRINT
30 PRINT"WHO IS YOUR FRIEND?"
31 PRINT"▩▩▩▩▩
32 PRINT
40 INPUT N$(2)
41 FOR K=1TO3000:NEXT
42 PRINT" ⊐
43 PRINTTAB(10)"HERE'S YOUR STORY..."
45 FOR I=1 TO 4 :PRINT:NEXT
50 FOR I=1 TO 5
60 READ A$(I)
70 NEXT I
80 DATA"WENT TO THE PARK.","WENT TO THE BEACH.","DROVE TO PARK AND SWAP."
85 DATA "FLEW TO CALIFORNIA.","WENT INTO A GIANT CAVE."
90 FOR I=1 TO 6
100 READ B$(I)
110 NEXT I
120 DATA "WHEN THEY GOT THERE","ON THE WAY" , "BEFORE THEY LEFT"
125 DATA "YESTERDAY MORNING","LAST NIGHT","AFTER SCHOOL"
130 FOR I= 1 TO 6
140 READ C$(I)
150 NEXT I
160 DATA" CLIMBED THE MOUNTAIN"," TRIPPED ON THE ROCK"," MISSED THE BUS"
165 DATA" SAW A BIG BEAUTIFUL BIRD"," SLID DOWN THE HILL"
166 DATA" ZOOMED DOWN A GIANT SLIDE"
170 PRINT N$(1);" AND ";N$(2)
172 PRINT
180 C= RND (TI)
190 C=INT (C*6)
195 IF C<1 GOTO 180
200 PRINT A$ (C)
202 PRINT
210 D=RND (TI)
220 D= INT(7*D)
225 IF D<1 GOTO 210
230 PRINT B$(D)
232 PRINT
240 E=RND (TI)
250 E= INT (7*E)
255 IF E<1 GOTO 240
256 H=RND (TI)
257 IF H<.5 THEN X$=N$(1)
258 IF H>.5 THEN X$=N$(2)
260 PRINT X$;C$(E)
```

```
262 PRINT
270 FOR I =1 TO 5
280 READ F$(I)
290 NEXT I
300 DATA "AND WAS REALLY AFRAID.","AND WAS QUITE EXCITED."
305 DATA"AND GIGGLED WITH EXCITEMENT.","AND SAID, 'WHAT A SURPRISE!'"
306 DATA "AND LAUGHED BECAUSE IT WAS FUNNY."
310 G= RND (TI)
320 G=INT(6*G)
330 IF G<1 GOTO 310
340 PRINT F$(G)
342 PRINT
345 GOSUB 500
350 FOR I = 1 TO 3:PRINT:NEXT
359 PRINT
360 PRINT"DO YOU WISH ANOTHER VERSION? (Y OR N)"
370 INPUT X$
371 PRINT"⊐
380 IF X$="N" GOTO 400
390 GOTO 45
400 STOP
500 FOR I = 1 TO 4
510 READ K$(I)
520 NEXT I
530 DATA"DID YOU SEE THE GIANT WORM?","DID YOU SEE THE DINOSAUR?"
540 DATA "DID YOU SEE THE INCREDIBLE HULK?","DID YOU SEE THE RATTLESNAKE?"
550 C1=RND(TI)
560 C1=INT(5*C1)
570 IF C1<1 GOTO 550
580 PRINT K$(C1)
582 PRINT
590 FOR I= 1 TO 5
600 READ L$(I)
610 NEXT I
620 DATA"IT LOOKED AWFULLY HUNGRY.","HE WAS VERY UGLY.","IT HAS A LONG NOSE."
630 DATA"IT WAS BLACK AND PURPLE AND GREEN.","ITS SHARP TEETH WERE SCARY."
650 C2=RND(TI)
660 C2=INT(6*C2)
670 IF C2<1 GOTO 650
680 PRINT L$(C2)
690 RESTORE
700 RETURN
999 END
READY.
```

A MODEL FOR TEACHER TRAINING WORKSHOPS

By John Losse and Rita Richards
Mathematics Department, Scottsdale Community College
9000 E. Chaparral, Scottsdale, Az 85253

OVERVIEW: The purpose of this presentation is to answer the question "So now that we've bought microcomputers, how can we train our teachers and staff to use them?'. The authors are experienced in training others on various computer systems, and recently conducted a 30 hour workshop in the use of the APPLE II which was attended by teachers with little or no computer experience and who represented disciplines as diverse as nursing, psychology, chemistry and music. Here is an outline of what the authors feel is an 'ideal' setting for teacher training.

PART I: WHAT FACILITIES DO WE NEED?

WORKSHOP SIZE: For a workshop of 15-20 participants we recommend a staff of two leaders and a technician. The leaders can alternate the presentation of new material with exercises to be carried out by the participants. During the time participants are working on the exercises, the leaders will be needed to circulate around the room to give help. A technician is vital to set the room up for each session and to keep the systems working properly, for it is an axiom that the systems most prone to 'problems' will unerringly seek out the users most easily frustrated by the same.

ONE PERSON ONE SYSTEM: Ideally, each participant should be seated in front of a microcomputer with plenty of space to spread out papers,manuals, diskettes and the other paraphernalia of creative work.

IF POSSIBLE USE A MICROVERTER! If your participant's systems use a TV display (as opposed to video monitor), they will be able to tune to the output of their own systems on one TV channel, and see what is on the leader's screen by switching to another channel (without destroying what is in their own computer's memory). This is an extemely useful facility because it enables the leader to teach by example, followed by a learner exercise. ('Watch what happens when I do this....Now you try this other') The device you need is called a Microverter and may be obtained from ATV Research for about $35.

OTHER HARDWARE AIDS: At various times you will need lots of blackboard space and an overhead projector. The overhead projector will be particularly useful for the display of program listings which, because of their length, do not fit on a single screen.

PART II: WHAT TOPICS TO COVER

FIRST SESSION - BASICS: Most of the participants will approach their first workshop session with a mixture of excitement and trepidation, so considerable time should be spent on attaining familiarity with the system itself: the keyboard, the TV screen, the printer and the disk drives. Participants should go through the mechanics of turning the system on and off, typing, correcting errors, cataloging a disk, running a saved program and initializing a blank diskette, until these routine tasks can be accomplished without difficulty. This is a good time to explain, in a general way, what a computer is, what the functions of the central processor, random access memory and long term memory are, and the place of peripheral input and output devices.

THE CONCEPT OF A PROGRAM - HOW MUCH PROGRAMMING NEED WE TEACH, IF ANY? While it is true that most users of computers today (airline reservation clerks, bank tellers, word processor operators, etc.) do not know any programming language, it is equally true that people who understand the rudiments of programming tend to make more effective and creative use of whatever hardware and software they have available. We think that, if at all possible, you should plan your teacher training workshop to include programming in some high - level language (BASIC, PASCAL, etc.) or possibly an 'authoring' language (one designed to facilitate the creation of lessons). In our most recent workshop approximately 60 % of the time was devoted to teaching BASIC, and the topics covered included GO TO, PRINT, INPUT, IF...THEN, FOR...NEXT, arrays, string variables and commands relating to graphics.

USE OF CANNED SOFTWARE: Once familiarity with the system is attained and as much programming facility as possible is acquired, the workshop sessions can begin to focus on the kinds of usage that will actually take place in the school. This can start with a smorgasboard of programs from various disciplines, chosen so as to display as much variety and creative pedagogical technique as possible. This is a good time to demonstrate programs of general interest such as word processing, classroom record keeping, and turorials in math, English, typing or reading. Then, depending on the homogeneity of the group, you may wish to break up into subgroups to look at programs in the various discipline areas. This is the time to bring in the software which the school owns and wishes to use in the classroom. It is vital to allow participants plenty of time to familiarize themselves with all the features of the software packages they will use before they hit the classroom. This is of course time consuming, and consideration should be given to letting the participants take the systems home if they wish. (They will tend to use them more, if only to be able to show their families and friends how to play video games, balance the checkbook, or use the word processor.)

PART III: OTHER CONSIDERATIONS

WHO SHOULD BE IN THE WORKSHOPS ? Motivation and interest are key factors. One cannot expect teachers who are dubious about the usefulness of microcomputers to leave the workshop as if they'd traveled the road to Damascus. It would be better to make skilled users of those who are really interested at first, and rely on them to supply, by example, the motivation that will bring others into the fold.

ARE FORMAL WORKSHOPS REALLY NEEDED AT ALL ? A district in Ohio has approached the issue by simply checking out its Atari's for a semester to interested teachers. The teachers took them home with the appropriate manuals and, by all accounts, came back a semester later ready to go into the classroom with them. This approach should be considered if you can afford to tie up your machines in this fashion.

INTENSIVE OR EXTENSIVE ? Between the 'Berlitz' approach of 'lets go all day every day until we get it' and the 'let's deal out the machines to the teachers and nature will take its course' approach, lies the intermediate route of assigning systems to participants for an extended period of time and conducting periodic group sessions to present necessary concepts, solve common problems, etc. In some respects this may be the best way to go about it, since it maximizes the time participants will actually spend with their systems while providing overall direction.

WHO SHOULD LEAD THE WORKSHOPS ? The first and most important requirement for leaders is thorough familiarity with the particular microcomputer system to be used. Next, leaders should be found who have some experience in the business of teaching computing, and finally, it is probably better to use peers of the participants if possible (fellow teachers, say, as opposed to district staff or somebody on loan from a computer store).

PART IV: CONCLUSION

Serious consideration should be given to the problem of how to train teachers and staff in the use of your microcomputer systems. What we have discussed here are the results of our own experience, that is, a workshop format which was successful for us and which we will follow in the future. The needs of individual institutions will of course shape their own training efforts. It is hoped that the observations made here will be of use to you.

PARENTAL INVOLVEMENT: THE KEY TO SUCCESS

Greg Wojtulewicz, Ed.D.

Alhambra School District
3001 W. Hazelwood
Phoenix, Arizona 85017

Introducing computer related projects within a school district often brings about various responses from parents. These responses range from the very positive (computers have a far reaching influence and may solve the basic skills problem) to the adamantly opposed (my child will never sit in front of a computer).

Those parents with positive attitudes present the district with a valuable resource. If effectively approached, their support can be limitless. Dealing with the negative attitude presents more of a challenge. These attitudes are often the result of a number of factors. Among the most prevalent are the lack of humanism, a feeling that the computer is too passive, and/or a fear based on uncertainty or a lack of knowledge. There is often a reluctance on the part of this group to become involved, even to examine that with which they are unfamiliar. This is much like the person who "doesn't like" a food which he has never tasted.

Perhaps the most effective way to reach this group is to provide concrete, hands-on experience. One method, therefore, would be to conduct computer awareness workshops for parents. The purpose of such workshops would be to primarily deal with these fearful parents, thereby lessening some of the negative attitudes.

159

Publicity for such workshops is essential and should be district wide. Their purpose should be made clear. Hours should be set at a time most convenient for the majority of parents. If the workshops cannot be sponsored by the school district itself, perhaps organizations such as PTA or community school may be urged to take on the responsibility. These workshops should be general in nature and not limited to programing instruction. Topics may include machine familiarization, demonstration of software for home, school, business, and career opportunities. To accomplish this it may be necessary to have competent volunteers serve as resource people to conduct certain sessions. Computer retailers are often willing to aid in such a manner. As interest develops, it may be possible to offer workshops with an in-depth study of specific areas. Some parents, because of related occupations, may have the expertise necessary to aid in conducting these workshops.

In addition to helping garner support from the parent who is uncertain or opposed to the computer's value, these workshops can also serve the parent who simply wants to bridge the "computer gap" with his child. This is the parent who wants a basic understanding of the machine and the terminology his son or daughter is using.

A further value of the workshops is to bring the computer out in the open. Going along with this, computers should be visible to the parents. This helps to "unlock" their mystery. At school open houses and conferences, for example, computers should be out and touchable. Simple programs welcoming a parent to the classroom can do much in the way of public relations.

The involvement of those parents who support the concept of computers in the classroom is essential. To have them assist with the computer project itself places the parent at the heart of the computer use in the school. Whether it be computer awareness or computer aided instruction, parents or other adults in the community can assist by working with students or caring for hardware or software.

Because the use of the computer in the classroom is relatively new, there appears to be a lack of organization both within education itself, and among the various hardware and software vendors. This problem provides unlimited possibilities for parental help. Parents, properly trained, can search for and review appropriate software by reading through the abundance of journals, advertising promotions, etc. They can write letters seeking information concerning anything from software to grants for computer projects. Properly informed, parents can knock on the doors of the community seeking all forms of local support. (This technique alone has funded several very successful computer projects.) Because it is a developing area, there is much room for research. While at this time federal grant money may be tight, some vendors are providing such support in the form of stipends and/or equipment.

At the school level, willing parents can be used as aides to type program material as it is developed by the teacher in the form of worksheets, etc. They can also copy software (when legal to do so) onto tape or disk. Many programs are printed in journals and need only to be entered on the computer and saved for use in the classroom.

As the program develops and expands new areas of need arise. Many times parents are able to help pinpoint these needs more accurately and can suggest further ways they can be of aid to the project.

In addition, parental help serves the very important function of freeing the teacher to perform his teaching duties without having to keep a vigil on students and equipment.

Because computer projects are expensive, it is not uncommon for those in charge of purse strings to have reservations. This is where that group of parents who exhibited a strong positive attitude can best help. Encourage them to be vocal in their support. It is a well-known fact that administrators will usually listen to parents. Therefore, urge parents to let their feelings be known to administrators, to the school board, and to other teachers. Encourage them to talk to neighbors, friends and other parents so the support foundation may grow and strengthen the computer project.

Never before has any technology entered the school and home so rapidly and caused such a stir. For example, a poll recently taken in a fifth grade classroom where computers are used revealed that one-half of the students had a personal computer or video game at home. Rarely have students and their parents had the opportunity to learn together about something that affects their daily lives. Much of what is written about computer use in the schools centers around students and staff. Very little includes parents. The obvious indication would be to take advantage of the situation and involve parents whenever possible, as part of the total computers in education picture.

SUGGESTIONS FOR PARENTAL INVOLVEMENT IN COMPUTER PROJECTS

COMPUTER AWARENESS WORKSHOPS FOR PARENTS

Effective publicity
Sponsorship
Objectives
Hands on materials
Time
Cost
Days

PRESENTATIONS FOR PARENTS

PTA/PTO meetings
Open House
School Board meetings
Science fair

PARENTS WITHIN THE COMPUTER PROJECT

Monitoring students
Caring for hardware and software
Cataloging software
Search for and review software
Write letters
Seek local support from business
Seek grants for funding
Type
Copy software

PARENTS WITHIN THE COMMUNITY

Urge support of others
Contact administrators
Contact school board members
Talk with neighbors, friends, other parents
Encourage attendance at workshops, presentations, etc.
Encourage other teachers to use computers

COMPUTER AWARENESS SEMINAR EVALUATION

Date: ________________

Are you the parent of a child in the Alhambra Schools? YES ___ NO ___

Have you had any previous computer experience? YES ___ NO ___

As an aid for planning future computer literacy seminars, your input is extremely important.

RATE THE QUALITY OF THIS SEMINAR AS FOLLOWS:

5	4	3	2	1
Excellent	Good	Okay	Fair	Poor

_____ 1. Topics selected for presentation
_____ 2. Clarity of concepts presented
_____ 3. Lectures
_____ 4. Demonstrations
_____ 5. Visuals
_____ 6. Handouts
_____ 7. Hands-on activities

RESPOND TO THE FOLLOWING OPEN-ENDED STATEMENTS:

8. After this seminar, my computer knowledge ____________________
____

9. Because of this seminar, I plan to ____________________
____

10. The content of this seminar ____________________
____

11. The most positive aspect of this seminar ____________________
____

14. My suggestions for future seminars ____________________
____
____

PLEASE GIVE A BRIEF STATEMENT THAT EXPRESSES YOUR FEELINGS ABOUT THE USE OF COMPUTERS IN THE SCHOOLS

____
____
____
____
____

IV. COMPUTER ASSISTED INSTRUCTION

Chemistry, Physics and Microcomputers

David Byrum
Globe High School
501 ASH St.
Globe, Arizona 85501

This workshop's purpose was to share with each of the participants some methods, materials and ideas for integrating microcomputers into science courses; in particular, chemistry and physics. At the conclusion of the workshop each participant should have been better able to understand the applications of micro's to the science curriculum as well as be in a better position to evaluate how micro's can fit into his particular school situation.

In other words, the purpose of this workshop was to help each participant know what to do after he learns to turn the microcomputer on and off.

COURSE ORGANIZATION and Getting Started

The first question most people ask when their school's microcomputers arrive is: "How am I going to use these things ?" Then in rapid succession, these questions arise: "What am I going to use them for in my classes ?", and "What am I going to do with them ?"

Fortunately, chemistry and physics are courses which, when taught with the appropriate organizational pattern, are easily adaptable to including micro's as part of the instructional scheme of things.

An appropriate organizational pattern is one in which the methods of instruction allow for flexibility on the part of the instructor and the student. The instructor is able to change how a lesson is presented without the change itself causing disorientation in the student's perception of the class. For the student, the appropriate pattern is a teaching method which helps train that student to be more flexible and open-ended in his thinking and class behavior patterns. If a student is accustomed to using a variety of learning methods, then the introduction of one more method will not be badly received.

What are some of the more commonly used methods of teaching science ? The following list seems to describe the most common methods.

A. Lecture-Lab - The "traditional" approach to science teaching.

B. Modular - A modified lecture-lab where The instructor sets up a block of subject material to be studied and within the time allowed, the student has some flexibility in deciding what activity will be done each day. Some days the instructor may lectrue, do a problem-solving group or work with individual students.

C. Self-Paced - The curriculum is divided into small self study units which must be mastered by the student one at a time. In addition, unit contains a set of learning objectives on which the student is tested.

For the modular and self-paced methods to be effective, each student must be given a complete set of Instructional Objectives for the course, and be tested only on those objectives.

Adding the microcomputer into the chemistry and physics curriculum at Globe High School was a simple matter since Physics is taught with the modular approach and Chemistry is a self-paced course. Appropriate computer lessons were inserted into the teaching units based on past experience of student problem areas. Some programs were made optional while others were made part of the course requirements by simply including the name of the computer program available to help teach a particular objective.

Implementing microcomputers into the science curriculum offers the instructor the opportunity to achieve many worthwhile goals. With the proliferation of computers in our society, using a computer in class creates in the student an awareness of computers which makes the course more relevant to the student. Using the micro also offers the teacher a chance for Great Teaching ! Using the computer frees the instructor from a role as a negative figure in drill & practice, grading etc. and lets him come closer to truly individualized instruction.

If the instructor is not afraid of change and new questions, or is willing to deal with these feelings, then the computer will fit in well in the classroom.

APPLICATIONS in the CLASSROOM

Uses of the microcomputer in the classroom is limited only by the imagination of the instructor and his students. Listed below are some more common uses.

1. Simulations - The computer is programmed to behave like some other system, e.g., a titration set-up, an Infrared spectrograph, a genetics experiment. These simulations allow the student insight into the workings of a new system, permit repetition of a difficult lab many times, and by the use of animation, simplify a conceptual relationship. They also demand active learning from the student as nothing happens until he participates.

2. C.A.I. - Computer Assited Instruction is a method in which the computer acts as a tutor, leading the student to learn some objective(s). Drill and practice is also included. With the computer's infinite patience, students can be given a random selection of problems and/or examples followed by immediate feedback of the results. C.A.I. can be highly motivating to underachievers and can significantly upgrade student performance.

3. Number Crunching - The computer can be programmed to check student's lab calculations and analyze their results. This includes graphing of data, determining % error and accuracy of observations. When the student is required to write the program which checks the results, by necessity, the student will thoroughly analyze the problem and carefully choose the steps leading to the solution.

In addition to these more common uses of the computer, the micro can also be interfaced to other lab equipment and then programmed to analyze the data. The lab equipment can range from a simple thermocouple to a video disk player.

Once the computer has received the data, it can analyze it in a variety of ways (graph, chart, etc.). This feature allows the student to see an analysis of an experiment as it is being done. Often, as a long experiment is being done, an equally long series of calculations must also be done. When this happens, the point of the experiment for many students is lost. By having the computer as an active participant in the doing of a lab, this problem of student confusion can be partially overcome.

For the instructor, the computer can become a gradebook, test grader and wordprocessor. Writing a lesson on the computer allows the teacher to store the completed pages of instruction as a file and then return to them the following year and modify them without having to retype the entire page. The only changes that are made are those which the teacher wishes to make. This ability to move words about lessens the clerical burden and seems to encourage more revision of course materials instead of just the reuse of existing lessons.

HINTS on PROGRAM WRITING

For a novice programmer, writing a program can become a maze. By following a few simple steps however, writing a computer program can become an easier task. These steps are:

1. Select a TOPIC that could use a computer presentation.
2. Decide what is is that you want the program TO DO.
3. ASSIGN the variables.
4. Write specific task SUBROUTINES.
5. Write the body of the program using the subroutines already written.

While these may seem like simple steps, it is amazing how many programs can be made very confusing because they were being written without any real forethought given to them.

Each of these steps will involve making sure that at each step of the program writing, proper organizational procedures are observed; resulting in a completed program in less than a week.

Chemistry, Physics and Microcomputers

SESSION HANDOUTS

Each of the session participants were given an outline of the sessions presentation, a resource list of software suppliers and resources for the classroom teacher, listings for three programs which with some modification can be run on micro's other than Globe H.S.'s PET's, and a list showing the computer programs currently being used in the Globe High School Science Dept.

SESSION ACTIVITIES

After the initial lecture presentation, the participants were able to go around to the PET computers set up in the room and run the programs available to demonstrate to themselves some of the types of lessons that the computer can present.

Each of the programs demonstrated are in current use at Globe High School.

1. PHYSICS Simulation - "GRAVITY", a program to study Newton's 1st Law.
2. PHYSICS Number Crunch- "Plot-a-Graph", a program which will accept up to 75 data points and plot an accurate graph either to the screen or the printer.
3. PHYSICS C.A.I. - "PC 1", a program to help learn to solve velocity and distance problems.
4. CHEM. Simulation - "TITRATION", a program in which a student can do a titration on the computer.
5. CHEM. Simulation - "IR Spectra", a program where a student can learn to identify organic compounds from a given infrared spectra generated by the computer.
6. CHEM. Lab Check - "QBARIUM", a program to check the results of a students stoichiometry experiment.

CREATING COMPUTER-ASSISTED INSTRUCTION WITH APPLE PILOT

by
Ruth A. Camuse
Arizona State University
Payne 47
Tempe, Arizona 85287

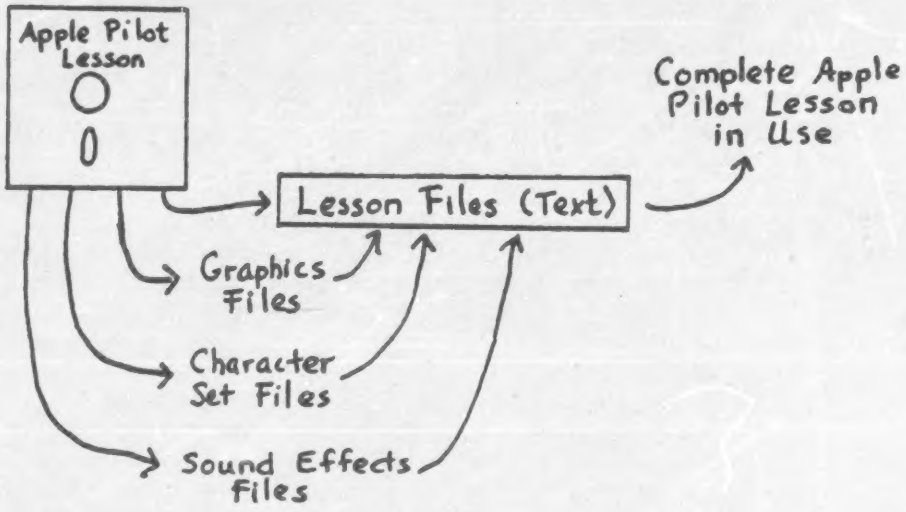
DESCRIPTION

PILOT is the name for a special programming language specifically designed with interactive style, computer-assisted instruction in mind. PILOT is available for several brands of microcomputers. The kind of PILOT to be discussed in this paper will be Apple PILOT, which can only be used with Apple microcomputers.

In order to be able to create programs using Apple Pilot, you will need a computer with at least 48K bytes of memory and two disk drives. You will also need the Pilot Author Diskette, the accompanying manuals, and an empty diskette which will be used to store the lessons you will be creating.(Once a lesson is created, students may run the lesson on a less expensively configured Apple with only one disk drive.)

The PILOT Author Diskette contains four different editors: the lesson text editor, the graphics editor, the character set editor, and the sound effects editor. You would use the lesson text editor to write the program itself, and the other editors are used to create graphic representations (pictures, maps, diagrams, etc.), music and other sound effects, and special characters, all of which you may wish to include within the CAI you are writing.

The following sketch shows the organization of a completed Apple PILOT lesson:



PROGRAMMING IN APPLE PILOT

Usually an Apple PILOT lesson is built using sections (or sub-divisions). Each section has its own name (label); each label must be no longer than six characters.

EXAMPLE:

```
*BEGIN
. . .
. . .
*PART1
. . .
. . .
*PART2
. . .
. . .
*TEST
. . .
. . .
*END
```

NOTE:Labels are optional. But most authors use labels, since they are a very convenient way to direct control of a program to a particular part of the lesson.

Like any programming language, PILOT has basic commands that must be learned. The major commands are listed below. The reader should refer to the Apple PILOT manuals for more details about the commands.

R:	REMARK. This is used exactly like a REM statement in the BASIC language. Anything typed after a R: statement will not show up when the student runs the program.
T:	TYPE. Whatever appears after the T: will be on the student's screen when s/he is running the program.
A:	ACCEPT. Waits for the student's response to a question, and temporarily stores in memory. But if a string or numeric variable is placed after the A: command (examples: `A:$X$` `A:#N3`), then the student's answer will be stored in memory under the name of the variable (`X$` or N3) and can be used throughout the lesson.
M:	MATCH. This command allows the programmer to list one or more responses that s/he considers to be correct. Format `M:Yes!Y!Yep`
J:	JUMP. Example `J:PART2` This command indicates that the next statement of the program to be performed will be first line in the section of the program labeled *PART2 . The control of the program will move on sequentially from there. (This command is like a GO TO statement in BASIC.)

C:	COMPUTE. Example: C:X=A*B In this example, A is multiplied by B and the result is set equal to the variable "X".
U:	USE. This command is used similarly to the GOSUB statement in the BASIC LANGUAGE. <u>Example</u>: U:REVIEW In this example, control jumps to the section of the program called *REVIEW , and goes on sequentially from there until an E: statement is encountered [This is sometimes referred to as "performing a subroutine".] Then control will jump back to the next statement following the original U: statement.
E:	END. This command is used to indicate the end of a program or subroutine.

Next a few of Apple PILOT's <u>conditioners</u> will be introduced. Conditioners follow the basic command letter (example-- TY:Hi! -- In this example, the Y is the conditioner). A conditioner sets up a test, and then performs according to the results of that test. It is possible to use more than one conditioner at a time.

Y	(Yes conditioner) Means → execute this statement if last answer was correct (was a successful "match" to the list of allowable responses indicated in the previous M: statement.
N	(No conditioner). Means → execute this statement only if last answer <u>did not</u> match any answers in the list of allowable responses indicated in the previous M: statement.
1 to 9	(Answer-count conditioner). Allows you to print different responses (text) on the student's screen, or jump to different places in the lesson, according to how many times he has answered the same question.

Several <u>modifiers</u>. are also available. One that would be <u>commonly used</u> is the H modifer.

hang

EXAMPLE:

```
TH: Your name?

A:
```

After the TH: sentence shows on the computer screen as the lesson is run by the student, the cursor will wait at the end of the sentence,next to the question mark, waiting for the student's answer. (Without the H modifer, the cursor hops down to the next display line, left-hand side.)

(3)   *TEST

```
PR:U                [This statement converts all student
                     answers to upper-case letters.  This
                     means that the student can type upper
                     or lower case letters in his answer,
                     and it will still be able to match
                     what is contained within the M:
                     statement.]

TH:What is the name of the 48th state to join
:the U.S.A.?
                    [Remember what the  H  modfier does when
                     the lesson is run?  Where will the
                     cursor wait for the student's answer?]

A:

M:ARIZONA           [The only acceptable answer.]

TY:That's correct.  [If the student has answered "Arizona"
                     the message "That's correct will print
                     out on the screen.]

TN:No, try again.
                    [If the student's answer did not match
                     "Arizona", the message "No, try again"
                     will print out on the screen]
JN:TEST             [Again, if the student's answer did not
                     match "Arizona", control of the program
                     will loop back to  *TEST.  The question
                     will then be asked again; the student
                     will answer again.]
```

Looking at some simple PILOT programs should help you to become better acquainted with the language.

```
(1)   *START           [Label for section of the lesson]

      T:What is the name of a yellow citrus fruit?
                       [This message will print out on the
                        student's scrren when s/he runs the lesson.]

      A:               [Waits for the student's answer.]

      M:lemon!grapefruit!grapefroot!
                       [Match statement]

      JY:TEST          [If the student has correctly "matched"
                        one of the possible answers shown in the
                        M:  statement, then control of the program
                        will JUMP to the section of the program
                        labeled TEST.]


(2)   *PREP

      T:Does 2 + 2 = 5 ?

      A:

      M:No!Nope!Untrue!False!

      TY:Good, that's correct   [If the student's answer matched
                                 one of the four  possibilities
                                 in the M: statement, then and
                                 only then would the text "Good,
                                 that's correct" appear on the
                                 student's screen.]
```

(4)

```
*MOOD
PR:U
T:Are you happy or sad today?

A:

M:HAPPY

TY:I'm glad to hear it!       [This message prints out only if
                               the student has answered "Happy".]

JY:PART1                      [Control of the program will JUMP
                               to the section called  *PART 1,
                               if and only if the student has
                               answered "happy".]

M:SAD                         [If the student did not answer
                               "happy", then his answer will be
                               compared against this  M: statement]

TY:I wish you were feeling better.
TY:Let's see if we can wipe away those blues.
                              [If the student had answered "sad"
                               then these two sentences will appear
                               on the video screen.]

TN:But are you more on the "happy" side or the "sad" side?
:Please answer again.
                              [If the student did not answer
JN:MOOD                        "happy" or "sad", then the above
                               text will print out on the video
*PART1                         screen;  then  the  JN:MOOD
                               statement will cause control of the
. . .                          program to  JUMP back to the
. . .                          beginning of the  *MOOD section of
                               the program.]
```

Another kind of conditioner is given the name of "expression". For example: `J(Q>12):PART4`

In this example, control would JUMP to the section of the program named "PART4" if and only if the condition (Q> 12) were true. If that condition was not met, then control of the program would proceed sequentially.

(5)

```
T: How old are you? (Answer with numerals, please!)
A:  #Q
J(Q>12):PART3
*PART2
...
...
*PART3
...
```

Sometimes, two statements of this type would appear in a program next to each other:

```
(6)     T(R>41):You are middle-aged.
        J(R>41):MIDDLE
```

These statements indicate that if R (the person's age) is greater than 41, then the statement "You are middle-aged" will be printed, <u>and</u> control will jump to the section named "MIDDLE".

It is possible to use more complicated conditioners, like `T((A<4)&(X=Y)):Right!` If A is less than 4 <u>and</u> X is equal to Y, then the statement "Right!" will be printed on the student's screen.

Finally, one more sample program will be shown. This program uses ANSWER COUNT CONDITIONERS.

```
(7)     *BEGIN
        PR:U
        T:What was the 48th state to join the U.S.A.?
        A:
        M:ARIZONA
        TY:That's Right.          [If the student has answered
                                   "Arizona", the message "That's
                                   Right." will appear on the video
                                   screen, no matter how many times
                                   s/he has answered the same
                                   question.]
        TY1:You got it on the first try!
                                  [This message will appear if
                                   and only if the student answered
                                   Arizona on the very first try.]
        TN4:No, Arizona was the 48th state to
        :join the U.S.A.
                                   [If student has answered
                                    incorrectly  for the 4th time,
                                    this message will print out
                                    on the video screen.
        JN4:NEXT                    . . . then control jumps to
                                          *NEXT.]
        TN:No, try again.          [Will print out after every
                                    incorrect answer, except after
                                    the 4th incorrect answer
                                    (the control of the program
                                    would not come down here after
                                    the 4th wrong answer, since it
                                    has already jumped to *NEXT.)]
        JN:BEGIN                   [Loops control back to
                                    beginning of this section. Note
                                    that this happens only if the
                                    student answered incorrectly
        *NEXT                       during the first three attempts.]
        . . .
```

The computer automatically keeps a count of how many times the student has answered the most recent question, and stores it in a place in memory called %A. Every time the computer comes upon an answer count conditioner (such as TY4), it compares the conditioner (4) to the number count the computer has stored in %A. If %A is found to be equal to the conditioner (in this case, 4), then the command "TY4:" is executed.

Obviously, it is not within the realm of this paper to describe all of the features possible when programming with Apple Pilot. I will list some of the most useful possibilities:

1. You may use the MATCH statement to accept varying spellings of the same word:

   CAP*T*L (will accept CAPITOL, CAPITAL, CAPOTAL, etc.)

   The * is called the "wild-card" symbol.

   If you want to be able to accept a long answer, you have the ability to write this match statement:

   M:Chewing&Bedpost&Gum

   [this match statement will look for these three words in the student's answer and count the answer as a correct match if it does contain these three words.]

2. LINK COMMAND. The command L: can be used to link separate lessons together.

3. RESPONSE TIMING. There is a simple way to create lessons which allow the student only a certain amount of time to answer a question.

4. RESPONSE EDITING. Certain statements inserted within a program will allow a student's response to be accepted and counted as correct, even though he may have typed a couple extra spaces, or forgotten to capitalize something.

5. Some of the same arithmetic functions you have in the BASIC language are available in PILOT.

   EXAMPLES: SGN(X) ABS(X) INT(X) RND(X) SQR(X)

6. To pull in designs from your GRAPHICS FILES, CHARACTER SET FILES, or music from your SOUND EFFECTS files, place these commands in appropriate places within the lesson:

   GX:MAP [Where MAP is the name of a map you have created with the Graphics Editor.]

   SX:Song TX:Symbols

7. The WAIT command. The command W:8 , for example, would cause the program to pause 8 seconds before going on. Useful if you want a pause between the printing out of two paragraphs or sentences, so that the student can read and assimilate meaning of what s/he's reading.

8. G:ES When this statement is inserted into a program, anything appearing on the video display screen will disappear (when the lesson is being run), and the next textual or graphic material in the lesson will appear. This causes a "screen-by-screen" display style, rather than "scrolled" style. (Scrolled: best described by referring to the titles that appear at the end of a movie. Some names disappear off the top of the screen display as more names appear at the bottom of the screen display.)

USING THE 4 EDITORS

A comfortable grasp of the operation of the four editors should be gained before attempting serious programming. The guide which follows will introduce the beginner to some of the most important elements of operation. Far more is possible with Apple PILOT than is explained here. Consult the Apple PILOT Editors Manual for more information.

Don't expect to create a complicated lesson by using this guide. The object of the following material is only for first acquaintance with the editors.

TO START UP:

Put the disk labeled AUTHOR in Disk Drive #1. The Lesson Disk (which is included in the Apple PILOT package) should be placed in Disk Drive #2. Turn on the computer!

The following MAIN MENU will appear:

```
Lesson text editor. . . . . . . . L
Character set editor  . . . . . . C
Graphics editor . . . . . . . . . G
Sound effects editor. . . . . . . S
Initialize a diskette . . . . . . I
Duplicate a lesson diskette . . . D

==>
Type option letter and press return.
Press  CTRL-C to return to this menu.
```

LESSON TEXT EDITOR:

We will be entering the Lesson Text Editor first, so press L , then press the RET key.

You will then see the following display:

```
            LESSON TEXT EDITOR

Based on  UCSD Pascal Editor

CALCULATOR   HELLO     IMMEDIATE
LEGENDS      MATCH


New,Edit,Run,Print,Delete,Quit
==>
Type option letter and press RETURN.
225 file blocks used, 55 remaining.
```

Type N for New, then press RET. The program will now ask you to name the lesson you'll be creating. Give the lesson a name no longer than 10 characters (stick with the 26 letters of the alphabet).

After you've named the lesson, you will see a screen that is almost blank, except for prompts showing on the top of the screen:

```
EDIT: CPY,DEL,FIND,INS,JUMP,REP,XCH,QUIT
```

Since you are creating a new lesson, use the INSERT command to begin typing in your lesson - - press the I key.

Now you are ready to enter a simple program. Try using the T: command to enter text which is more than one line long. Note that there is a very useful feature called automatic formatting. This means that a paragraph of up to 250 characters long may be typed after a single T: command, with no need for you to press the RETURN key. Just keep typing away; the editor will make sure no word is divided in half at the end of a line.

```
T:Plus, when editing your lesson later,
:automatic formatting will allow you to
:add or delete characters, words, and
:sentences without worrying about format
:exceeding margins, or leaving blank
:spots within sentences.
```

Try typing a few lines after a T: command, using the automatic formatting feature. If you make a mistake use the ← key to backspace and then type over errors.

By now it is likely that you've made at least one mistake that could not be easily corrected by using the ← key. How will you edit that mistake?

First you must leave the INSERT mode by asking the computer to accept what you've just typed in. Do that by pressing the ESC key. [Note: you can leave modes within the editor without saving any changes you've made by pressing the CTRL and C keys at the same time.]

Then you should move the cursor to the location of the mistake. This can be accomplished for travels of only a few letters or words by using the ← and → keys.

Some other ways to move the cursor:

(1) Return key

(2) Space bar

(3) Press CTRL and O at the same time to go up one line.

(4) Press CTRL and L at the same time to down one line.

(5) Press CTRL and I at the same time to move the cursor eight spaces at a time.

(6) Use the P editing command to move a "page" worth of distance.

(7) Use the J editing command to JUMP to other parts of the lesson (Beginning, End, to a particular label, etc.)

After you've moved the cursor to the spot where the mistake was made, use the various editing commands (prompts appear on the top of the editing screen) to accomplish any editing you wish to do. The most commonly used editing modes are INSERT [INS], DELETE [DEL], and EXCHANGE [XCH].

(1) INSERT mode - Press I ; then type in the character, word, or sentence you wish to insert.

Press ESC to tell the computer to accept the change you just made.

Press CTRL & C at the same time to leave the insert mode without accepting any changes you've just made.

Don't be dismayed that words move across the screen and even seem to disappear! This is normal. When you do your insertion, and then press ESC, the text will once again appear in a normal fashion.*

(2) DELETE mode - Press D ; then use the ← and → keys to delete characters and words.

You can delete entire lines of text by using the RET key.

Press ESC to tell the computer to accept the deletion you just made.

Press CTRL & C at the same time to leave the deletion mode without accepting any changes you've just made.

---

*Some insertions are more difficult to make. Consult pp. 49-51 in the Apple Pilot Editors Manual for details.

(3) EXCHANGE mode - press X. This is best explained through an example. Say you want to change shew to show. You have previously moved the cursor to the "e" in shew. You pressed X to enter the exchange mode. Now type an "o" to replace the "e". Press ESC to tell the computer you wish to accept the change you just made. The word will now be changed to "show".

Again, you may leave the EXCHANGE mode without saving the changes you made by pressing CTRL & C at the same time.

NOTE: You may only exchange words of the same length. If you want to change "he" to "they", for example, you can only change "he" to "th". Then accept by pressing ESC. Then use the INSERT command to add the "ey".

Now some hints about capitalizing. As you may know, the Apple II and II Plus computers do not innately possess lower-case letters. Apple Pilot does! However, lower case letters will only appear in the lesson when it is being run. While you are in the lesson text editor, everything will be capitalized. A true capital letter is made by using something called "Inverse Video" (black letter on white background).

This is how to make sure a letter will be capitalized when the student runs a lesson:

(1) Make sure you are in the INSERT or EXCHANGE modes.

(2) Capitalize words in either of the following ways --

Press the CTRL and Z keys at the same time, and the next letter only will be capitalized. You will know it is going to be capitalized when the student runs the lesson, because the letter will appear in Inverse Video.

Press the CTRL and A keys at the same time, and all subsequently typed letters will be capitalized. To make letters all lower-case once again, press CTRL and A at the same time.

After you've tried out the editor and gained some degree of familiarity with its operation, leave by pressing Q for quit. Then a message similar to this one will appear on the screen:

Save this lesson as FIRST?

(1) If you wish to save your work under the name you originally gave when you first entered the lesson text editor, type Y (for Yes), then press RET. This will return you to the Lesson Text Editor MENU. If you want to run the lesson to see how it would appear to a student when s/he runs the lesson, select the RUN option from the Lesson Text Editor Menu.

(2) If you wish to save the lesson under another name, type N (for No), and you will be given the opportunity to rename the lesson.

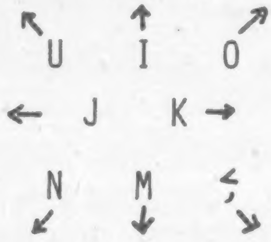
(3) If you wish to get out of the lesson text editor without saving the lesson you were working on, don't type anything, just press RET. Then this message will appear: SAVE as what name?
Press RET again. You will be returned to the main menu, and your lesson will have been erased.

Now go ahead and play around with the Lesson Text Editor. Try a few simple programs. Run them in the Lesson Mode through the Main Menu of the Lesson Text Editor. During the running of a lesson, sometimes the lesson will come to a stop because of a programming error. If this happens, you can press RET key or the space bar, and the lesson will continue.

THE GRAPHICS EDITOR

From the MAIN MENU, press the G key, then RET to enter the Graphics Editor. A list of graphics files will appear. Since you will be creating a new graphic (picture), press N for New, then press RET. Name your graphic, then press RET.

You will immediately be confronted with a blank screen. Only a tiny blinking light is present. Don't panic! The light is there to indicate where you will begin drawing. You can draw by two methods--game paddles (if you have them) or by using certain keys on the keyboard. These are the direction keys which are used to move the little blinking light (cursor "rubber band") to draw lines, circles, etc.



Pressing the I key, for example, will make the cursor move upwards. Press I a few times. Doesn't move the cursor very far, does it? If you want to move further faster, hold the REP (Repeat) key down at the same time you hold down one of the direction keys.

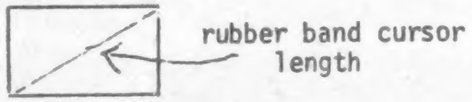
Experiment with the keys a little bit. Use the J key. Then the U key. See what happens.

But you still have not drawn anything. To do that, you may use the following commands:

(1) D Draws a line from where you first found the little blinking cursor to where you expanded the "rubber band cursor".

(2) S Skips to the spot where you moved the cursor, without drawing a line.

(3) F If you draw a diagonal "rubber band" cursor with the direction keys, press F to make an empty box (Frame).

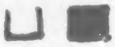


(4) B If you draw a diagonal "rubber band cursor" with the direction keys, press B to make a solid box.

(5) C Make a circle or oval by making a diagonal "rubber band cursor", then pressing C.

(6) A You can fill a partially or completely enclosed area by moving the cursor inside that area (Use the direction keys, then the SKIP command). Then press A , and the area will fill.

Example :



(7) E This command will erase the last line, circle, etc. that you drew.

(8) You can choose 9 different color options. Just press the appropriate key to choose the color you want. Then keep drawing.

KEY	COLOR	KEY	COLOR
0	Black1	5	Orange
1	Green	6	Blue
2	Violet	7	White2
3	White1	8	Reverse
4	Black2		

(9) T Typing text alongside a design can be very useful for maps, diagrams, etc. First move the cursor to where you wish to type text (use direction keys). Press S for Skip. Then press T. Type whatever words you wish to appear. Press ESC to tell the computer to accept your line of text and go back to the regular drawing mode. Then press D if you wish to underline the text; press S if you do not wish to underline the text.

(1C) You may PRINT any graphics design you have created. You should have an Apple Silenttype Printer. Hold down the CTRL key while you also press the P key. Whatever design was present on the video display screen will now be printed on paper.

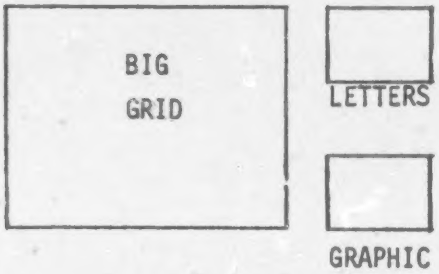
Play around within the Graphics Editor, getting used to the various commands. Switch colors. Try to make boxes and circles. Use the area fill command. When you've become somewhat comfortable with the Graphics Editor, leave by pressing Q for quit.

THE CHARACTER SET EDITOR

Enter the Character Set Editor from the MAIN MENU by pressing C then RET.

It is useful to view an already created character set in order to begin to understand this editor. So when the next display appears asking you what you want to do (the Character Set Editor MENU), type E for edit old file. Then ask to see the file called ASCII.

The next display you will see looks rather confusing! There are three boxes.



You are in what is called the letter box mode. Type the letter "a". The letter "a" will appear in the Big Grid, in the same position within the grid (top left-hand corner) as the "a" in the letter box. The Big Grid is really a magnifier, allowing you to see what little tiny dots in the video display are filled in so that the image of "a" is created.

Now type "b", "c", "d", etc., and watch the Big Grid.

NOTE: To move around in the Letter Box, you can use the ← key (to erase), the → key and RET key. Do not use the space bar.

Now that you have an overview of the character set editor, leave the ASCII character set by pressing ESC, then Q.

The message "Save this character set as ASCII?" will appear. Press Y for yes. (Important Note: If you ever go into the ASCII file again and change the construction of the ASCII characters, don't press Y for Yes. If you do press Y, the original "good" character set will be replaced with your changes!) See pp. 28-30 of the Apple Pilot Editors Manual for more details about naming and saving files.

Now you will again see the Character Set Menu. This time select to make a new character set. Name it DIAMOND. Then you'll see the Character Set Editor Screen again.

Let's say you want to create a special symbol that looks like this: ◇ We'll let the letters di
am
stand for ◇. So put di in the letter box by typing
am
di (then press return key) am .

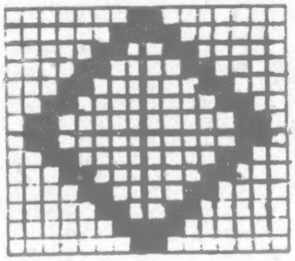
Press ESC to tell the computer to accept the letters you just typed into the Letters Box. This also puts you into the Big Grid Mode. Notice a little blinking light in the upper left-hand corner of the screen. That is the cursor. You can move the cursor by using the direction keys or paddles.

↖ ↑ ↗
U I O

← J K →

N M <,
↙ ↓ ↘

The 4 upper left-hand boxes of the Big Grid correspond to the "di
am"
you typed into the Letter Box. You will be filling the tiny squares within those 4 boxes in the BIG GRID to create your special character (or picture). After you are finished creating the picture, you will be able to use the code "di
am"
in order to show the picture of the diamond during any lesson. Use this grid paper design of a diamond to help you create your own "special character".



The 4 upper left-hand boxes of the BIG GRID

(Next page tells how to fill the squares.)

When you have moved the cursor by using the direction keys, and you have reached a box you want to fill in, press F for fill.

Press F for fill.

Press E for erase.

Press space bar. You can now use the direction keys to move around freely without filling or erasing.

Press L to return to the Letter Box Mode.

Special Notes on the Character Set Editor:

(1) As you have seen, it is possible to use a combination of letters in the letter box to create a picture much larger than the size of a normal letter. Example:

FLO
WER

You would use 6 boxes within the BIG GRID to create a picture of a flower which would be 6 times as big as a normal character. After you've designed the flower using the 6 boxes in the BIG GRID, the GRAPHIC box in the lower right-hand corner of the screen will show you how big the flower will be if you use it in a lesson.

(2) It is necessary to keep a record of the letter or letter combination you used to create a special character or picture.

(3) It is possible to PRINT from the character set editor (if you have an Apple Silentype Printer). Make sure you're in the BIG GRID mode. (If you're in the Letter Box Mode, press ESC.) Press the CTRL and P keys at the same time, and the Character Set Editor screen contents will print out on paper.

(4) Refer to the two Apple Pilot Manuals for further information on how to create character set graphics, and how to integrate them into lessons you're writing.

When you have familiarized yourself with the Character Set Editor, press Q for Quit. The message "Save this character set as DIAMOND?" will appear. Answer Y for Yes, then press RETURN.

Now that you have created a diamond symbol, you may insert the diamond into any lesson you are creating with Apple Pilot. At an appropriate place, insert the following statements into a program you are creating in the Lesson Text Editor:

```
TX:DIAMOND

T:        DI
T:        AM
```

When you run a lesson containing these statements, a picture of a diamond will appear!

THE SOUND EFFECTS EDITOR

You may enter the sound effects editor from the MAIN MENU by typing S, then RET. You will see that there are already some songs on file. You can play whatever sound file you wish by pressing E for edit, then indicating the file name of the song. Then press P for play.

After you've heard a little concert, let's familiarize ourselves with the Sound Effects Editor by creating a new sound file. Press N for New. Then name your file (10 characters or less). Then this message will appear:

INSERT DELETE PLAY QUIT ?(HELP)

Press I for Insert. This message will appear:

ESC to accept, CTRL-C to abort

NOTE

1 4C 1/4

blinking cursor

dummy note

Press RET to accept the dummy note as the first note of your composition. Do this several times. You will hear the note played every time you press Return.

Now press CTRL and C at the same time to get rid of what you've just done, and start all over again. Press I again.

Say you want to replace the "dummy note" with

number of note → 1 3E 1/2 ← length of duration

octave

tone

Press 3 , then E , then " , Then press RET .

The note will now be changed. From now on the 3E 1/2 will be the dummy note until you change it to another note. When entering musical compositions which tend to have a lot of repeated notes, this is a real time saver.

You will need to refer the the Apple Pilot Editors Manual for much more detailed information about the Sound Effects Editor. In the meanwhile, you can use the mini-help information on the bottom of the screen to help you create your own musical composition. Or--enter this information to write "Mary Had a Little Lamb".

1	3E	1/2	(You have already entered this note)
2	3D	1/2	(Press D , then RET)
3	3C	1/2	(Press C , then RET)
4	3D	1/2	(Press D , then RET)
5	3E	1/2	(Press E , then RET)
6	3E	1/2	(Press RET)
7	3E	1	(Press ! , then RET)
8	3D	1/2	(Press D , then RET)
9	3D	1/2	(Press RET)
10	3D	1	(Press ! , then RET)
11	3E	1/2	(Press E , then " , then RET)
12	3G	1/2	(Press G , then RET)
13	3G	1	(Press ! , then RET)
14	3E	1/2	(Press E , then ", then RET)
15	3D	1/2	(Press D , then RET)
16	3C	1/2	(Press C , then RET)
17	3D	1/2	(Press D , then RET)
18	3E	1/2	ETC.
19	3E	1/2	
20	3E	1/2	
21	3E	1/2	
22	3D	1/2	
23	3D	1/2	
24	3E	1/2	
25	3D	1/2	

26	3C	1
27	4C	1

When you are finished composing the song, press ESC to indicate you are finished. You may then play the song by pressing P.

To EDIT MISTAKES:

There are only two editing modes: delete and insert. The first step you would take to edit a mistake is to move the cursor to the spot that needs to be edited. To move the cursor use the ← and → keys.

Once you have moved the cursor, get into the delete or insert mode:

(1) DELETE mode. Press D for Delete. Pressing RET key will delete the note located by the blinking cursor. Press ESC to tell the computer to accept the change you just made.

(2) INSERT mode. Press I for Insert. Then change the line located next to the cursor as if it were a "dummy note" (as when you first created the composition). Press RET. Then press ESC to tell the computer to accept the change you just made.

(3) There is no EXCHANGE or REPLACE editing mode in the Sound Effects Editor. In order to replace a wrong note, you must first delete the note, then insert the correct note.

Designing Microcomputer Software in K-8 Mathematics

James H. Wiebe, Ph.D.
Department of Curriculum and Instruction
The University of Southwestern Louisiana
Lafayette, Louisiana 70508

The microcomputer has the potential to revolutionize the classroom like no other tool since the textbook. Possible uses of this wondrous device have been well documented by others (e.g., Moursund, 1980). This paper is about one of these uses--computer assisted instruction (CAI) in the area of mathematics.

Those who have spent time sifting through the numerous offerings of would-be software developers in mathematics CAI are well aware of the fact that it is very difficult to find much of value among them. Most of what is available does not make use of the tremendous capabilities for interactive instruction, graphics, diagnosis of student responses, or branching capabilities, let alone knowledge of how children learn mathematics. There is a need for software in elementary mathematics that is based on sound principles of cognition, that takes into consideration the realities of the classroom situation and the needs of typical teachers, that incorporates recent research findings about the way children learn mathematics, and that makes full use of the power of the microcomputer but at the same time considers its limitations.

Before going further, we need to clarify what we mean by the term "microcomputer." Most educated people in our society have some notion of what a microcomputer is--a small version of "the real thing" that they have seen in large business establishments, banks, universities, or on television. For purposes of this paper, a microcomputer will be defined as a small, self-contained computer (desk-top size or smaller) for use by one individual at a time.

Equipment Considerations

Although it is difficult to obtain any statistics from manufacturers, it is safe to state that the vast majority of microcomputers being purchased for use in elementary and secondary schools are either PETs, TRS-80s, or Apples with 4K or 16K of memory and with a cassette tape recorder for program storage. Certainly, many schools have purchased one or more of these brands with much greater memory capacity, with a printer, and/or with mini disk systems, but these configurations are the exceptions to the rule. Thus, when designing software for microcomputers that has the chance of being widely used, one must consider limited memory size, slow loading of programs via cassette tape, and probably lack of color, sound and other devices like light pens and "paddles."

Microcomputers also have certain advantages over large computers that should be considered when creating educational software. PETs, TRS-80s, Apples, Ataris, and IBM Personal Computers all have excellent graphics capabilities. This makes it possible to present pictorial models, game boards, graphic illustrations, and graphic rewards to the learner. Another feature of the type of microcomputer about which we are talking is the use of the cassette recorder(controlled by the computer) to deliver prerecorded instructions or audio drill material (Kehrberg, 1981).

Features common to all computers that should be considered when developing educational software are (1) the ability to compare a response input by the student to a number of possibilities and to branch to an appropriate remedial sequence, (2) the use of timing loops in displaying text or graphics, (3) the ability to store, classify and output records of student input, and (4) random generation of numbers and/or selection of problems. All of these are desirable features of microcomputer software as will be discussed below in more detail.

When developing microcomputer software in mathematics, in most cases it is advisable to stay with programs that fit into 16K of memory (these can always be stored on disk at a later date), that do not make use of peripheral devices, and that are written for one or more of the "big three" microcomputers. It is also advised that you take advantage of the branching and graphic capabilities of these machines, and, where appropriate, the possibilities for recorded voice. There are certainly many possible exceptions to this advice, especially if you are aiming at a limited market, such as Apple Computers with disk drives or if the type of usage of your software requires disk storage. An example of this is microcomputer diagnosis in mathematics--for the diagnosis to be useful to the teacher, data about the student should be stored and compiled into a report, a task which would be difficult using cassette tape.

Design Considerations

For those who have taken a course or two in computer programming but have spent little time developing elaborate software, the temptation is, after coming up with an idea for some software they would like to develop, to simply sit down at the computer and to start programming. This is a mistake! CAI programs are usually extremely complex and attempts to write the program without thorough planning will lead to frustration, wasted time, and programs that run poorly if at all. The first step in developing educational software, as with any software that will be used by others than the developer, is to thoroughly plan and design your product.

With the increasing importance of computers in many areas of our society, the field of software engineering has rapidly developed. Many of the findings in software design for business and industry are relevant to educational software development for microcomputers. According to Wasserman (1977), perhaps the most important step in software design is to gain a complete understanding of the problem you are designing the program for. This usually includes a clear definition of program requirements, followed by a series of refinements into more and more detailed statements of the problem or into smaller subtasks.

Suppose you wanted to provide upper elementary students with CAI in solving verbal mathematics problems. A general statement of the problem might be "the student will be given tutorial instruction and practice in solving verbal problems involving percentage increase and decrease." To clarify in your mind how to go about this task, you might decide that in the program you would have the student (1) identify which of several possibilities is a correct restatement of the problem, (2) identify a graphic representation of the problem, (3) input the order of operations used to compute the answer, (4) give an estimate of the answer, and (5) input the correct answer after pencil and paper computation. You might further refine the problem by stating the way incorrect and correct responses will be dealt with, how the student will control the program, and the types of rewards/reprimends students will get for correct/incorrect responses.

An important step in the design stage is to then decide how to program each of these small, comprehensible parts of the program. This might involve subprograms or a series of clearly identified subparts to the main program.

When designing software for use in mathematics, it is helpful to consider some of the possible types of educational software. Do you want your program to teach a concept, provide drill and practice, provide instruction in problem-solving procedures, does a combination of these or does something else? If this is not clarified before you begin programing, you may end up with a product that does none of these things well. The National Council of Teachers of Mathematics, in its publication, "Guidelines for evaluating computerized instructional materials" (Peck, Johnson & Kansky, 1981), lists several types of educational software, each requiring a somewhat different approach to design.

1. Tutorial. E.g., the student is led to discover the Pythagorean Theorem.
2. Drill and Practice. E.g., the student is given timed drill with single-digit products.

3. Simulation. E.g., the student is emperor of a country, must manage its finances, and is informed of the effects of his/her choices on the country.
4. Instructional gaming. E.g., when the student correctly answers multiplication problems, a "token" is moved on a gameboard displayed on the screen.
5. Demonstration. E.g., the computer shows graphs of algebraic equations and inequalities after the student inputs appropriate values.
6. Informational. E.g., the student enters a number and the computer outputs the prime factorization of that number or states that it is a prime number.
7. Instructional management. E.g., test generation, record keeping.

Of the first six possibilities, the microcomputer has the potential to be the most useful in 1 and 3. Yet, it is in these areas where the fewest quality programs have appeared. The probable reason is that these types of programs need to be developed by those who are very knowledgeable about classroom realities, learning theories, mathematics curriculum and research in education, whereas those who have been developing most microcomputer software are not knowledgeable in these areas.

Learning Considerations

When developing software that attempts to teach mathematical concepts or skills, there are several very important considerations. These are the learning theories and research findings that have had a major impact on mathematics instruction and curriculum over the past ten or more years.

First and perhaps most important is that instructional should be meaningful. After numerous studies, researchers have concluded that students will retain more, achieve higher scores on tests, and be able to transfer more to new situations if they understand what they are doing (Suydam and Weaver, 1975). Mindless memorization of rules, facts, or steps in skills or facts can be only partially successful when compared with teaching the same skills or facts meaningfully. Microcomputer software should stress mathematical relationships, principles, and meanings. For example, instruction in one-digit by two-digit multiplication, instead of only focusing on the steps of the algorithm, should be based on reference to real-world examples, on models, on the principle of repeated addition, and later on place value and the distributive property. See Figure 1.

```
 15
 15                                   15  x  6
 15    6 times      15              = (10 + 5) x 6
 15               x  6              = (10 x 6) + (5 x 6)
 15                 90              = 60 + 30
+15                                 = 90
 90
```

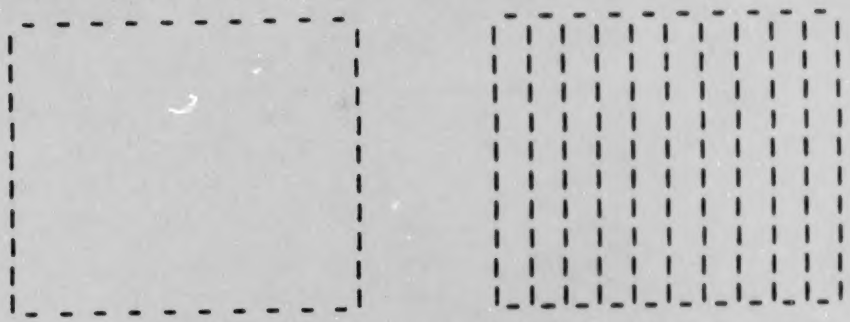
Repeated Addition   Distributive Method

Figure 1: Understanding the multiplication algorithm

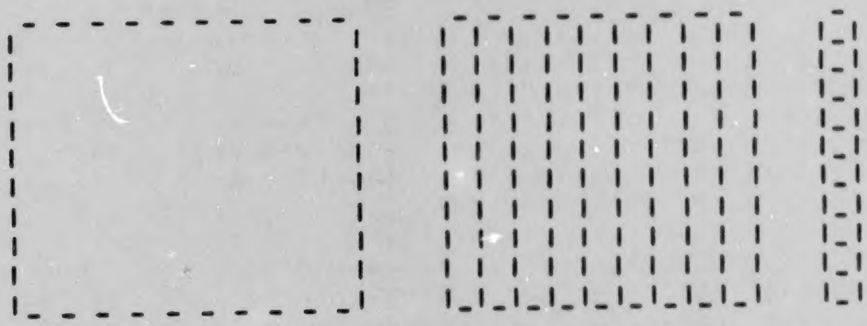
Another major consideration is that children in the elementary grades are, for the most part, in the stage of concrete operations which means that they either need to be able to use or visualize physical objects when dealing with mathematical abstractions. Multi-digit subtraction, for example, is not meaningful without some reference to real objects. Furthermore, recent research in the use of physical and pictorial models for mathematical symbols, abstractions, and operations has demonstrated that children tend to achieve more when models are used in instruction (Suydam and Higgins, 1977). Thus, whenever it is appropriate when developing tutorial software in elementary mathematics, ask the child to think of specific concrete experiences that relate to the concept you are trying to develop, ask the child to actually do the manipulations pictured on the screen (the materials needed should be clearly identified in the documentation of the software and at the beginning of the program), and/or use graphic representations of concrete models should be shown on the screen to help clarify symbolic manipulations or abstract discussions.

Figure 2 shows one way to illustrate the problem "200 - 37" using microcomputer graphics.

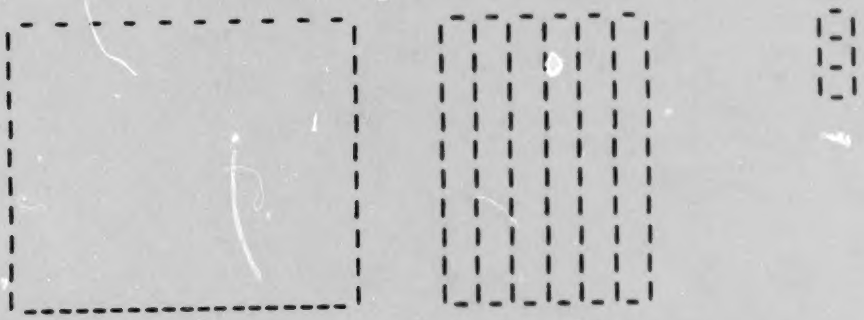
Frame 1 (Represent 200)



Frame 2 (Regroup one of the hundreds)



Frame 3 (Regroup one of the tens for 10 ones)



Frame 4 (Remove subtrahend, express answer)

Figure 2: Graphic development using pictorial models

Of course, with microcomputer graphics capabilities, one could simulate movement to illustrate the steps in regrouping (e.g., use timing loops and the slow "removal" of a hundred from the screen and its replacement by 10 tens) and subtraction (e.g., use the same technique to show the removal of 3 tens and 7 ones). This whole process could be interspersed with appropriate text and questions that lead the child to important discoveries.

Below is a list of mathematical areas and possible pictorial models that would be well suited for graphic manipulation on the microcomputer:

1. Number concepts--most any shape to show objects; circles or other regions to show groups.
2. Numeration and place value--numeration blocks (see Figure 2), tally marks, place value charts, abaci.
3. Operations on whole numbers--numeration blocks, place value charts.
4. Meaning of fractional numerals--rectangles, regular hexagons, etc. divided into congruent parts and appropriately shaded; number lines.
5. Decimal and percentage numerals--squares divided into 10, 100, 1000, etc. congruent parts and appropriately shaded; place value charts.
6. Operations on fractions, decimals and percentages--shaded rectangles divided into apropriate congruent parts and shaded, number lines.
7. Geometry--appropriate shapes.

Perhaps the best use of the microcomputer in teaching mathematics is the incorporation of discovery into teaching sequences. Ideally, in a classroom setting, children should discover as many mathematical principles and relationships as possible--research has shown that what children discover they will remember longer, be able to apply better and be better able to transfer to new learning situations than what is told to them and memorized (Riedesel & Burns, 1973). Realistically, however, it is very difficult to handle discovery approaches in a large class setting because the teacher cannot supply appropriate, individualized guidance to each and every child that will lead to the correct generalization. The microcomputer makes it possible to arrange the steps in guided discovery that are at the student's level, that contain as many or as few steps and hints as the learner needs, and that are appropriately paced.

An example of the discovery technique in mathematics would be the following instructional sequence relevant to the Pythagorean Theorem. The learner is presented with numerous right and non-right triangles. Each triangle is graphically presented, allowing the student to build squares on each of the sides. After several examples, the values are arranged in tabular form and the student is asked to examine the data and make an appropriate generalization. If he/she does not give the appropriate response, further examples are examined with additional hints (e.g., the graphic display could show the movement of the squares from the two legs of the right triangle to fit together to form the square on the hypotenuse). Again, the data are presented in tabular form and a generalization is asked for.

In their "An Agenda for Action: Recommendations for School Mathematics of the 1980s," the National Council of Teachers of Mathematics (1980) stated that the foremost goal of mathematics instruction in this decade should be problem solving and application of mathematical concepts and skills. Recent research findings have shown that certain strategies taught to students can significantly improve their problem-solving achievement (Kantowski, 1981). Some of these strategies are to (1) have the student restate the problem, (2) have the student draw a diagram, (3) represent data in the problem or in the solution in tabular form, (4) have the student designate the process that will be used before actually doing any computations, and (5) looking for patterns. Since problem solving is a very individual endeavor and since only a few of these strategies will be useful for a given problem, the microcomputer seems to be a natural vehicle for developing and sharpening these skills.

A final learner consideration is the level of difficulty of mathematical content presented to the student. When sitting before the computer, there are no grade designations to stigmatize the slow student or to hold back the advanced student. In any classroom, the ideal situation would be for each student to be learning at his or her level, not at an arbitrary level established by a textbook publisher, district minimum standards, or by the average level of a class of students. With the branching capabilities of computer programs, it is possible to find out what the student knows and does not know very quickly and adjust the level of problems accordingly. For example, suppose you are developing a lesson on verbal problem solving techniques. It would be appropriate to imbed in the program diagnosis of student responses and adjustment of computations accordingly so that computation problems do not detract from your main task--that of teaching the student to be a better problem solver.

There are many other possible learner variables such as preferred learning style, reading ability, typing ability, ability to visualize and comprehend graphic stimuli, competitiveness vs. desire to work cooperatively with others, locus of control, etc. that can be considered when developing microcomputer software. The effect of these variables on students' abilities to learn via microcomputer CAI should be explored through research.

Programming Considerations

The basics of software engineering outlined by Wasserman (1977) are applicable to some degree when writing microcomputer CAI software. They are as follows: (1) Use good internal documentation procedures. This is for your own benefit since you will invariably need to come back and make changes and in complex CAI programs, it may be difficult to remember what you were trying to do in a part of the program even the following day. (2) Use mnemonic variable names for easier tracing of program flow. (3) Minimize the use of GOTO statements --they make the program difficult to trace and understand. (4) Use a modular program structure, associated with the sub-parts of your program.

Aside from content, display of textual material is probably the most important consideration when writing CAI programs in any subject. The student will quickly lose interest if the text rolls off the screen before he/she has had a chance to read it or if the student does not know how to communicate a response or continue execution of the program. The following are some general suggestions for making your CAI programs useable and successful in the elementary setting.

1. Do not allow the text to roll off the top of the screen.
2. Display a limited amount of text and/or graphics on the screen at one time. In many cases, it may be appropriate to include only one or two sentences followed by an action by the learner (e.g., responding to a question, pointing to a portion of the screen, etc.).
3. Allow the user to control the display of text (Kehrberg, 1979). For example, display on the bottom of the screen in bold letters, <HIT RETURN TO CONTINUE>. Timing loops should not be used alone to control the length of time text and/or graphics that are integral to the meaning of the instructional are displayed--each "frame" should be controlable by the learner. Another way to allow the learner to control the display is by answering questions--when the answer to the question is entered, the program goes to the next textual display. Finally, standardize your method of indicating that the material has been read by the learner (Kehrberg, 1979). For example, always have the student hit "return" to continue.
4. Make sure that reinforcement for correct responses is more motivational than that for incorrect responses (Kehrberg, 1979). For example, only display a very small, "sorry, that was not correct" for each incorrect response, but for each new correct response, display an increasingly large, more elaborate graphic. One possibility would be to ask the child to see if he can "burst the balloon." Then, each time he/she enters a correct response, show a graphic of a man with a balloon that gets increasingly bigger. After a given number of correct responses the balloon bursts with an appropriate visual display.
5. In many cases it is appropriate to both a timing loop and student control of displays. According to Kehrberg (1979), when several displays are similar in appearance, students may be looking away from the

screen when there is a change in display and will hit "RETURN" to go on to the next display, missing important text.

6. Methods for stopping and continuing the program and for entering answers should be made clear to the user (Kehrberg, 1979). You may include in the program a method for the student to get further instructions anytime he/she is not sure what to do (e.g., "help").

7. Anticipate inadvertent (or deliberate) use of control keys during execution of the program. You may wish to deactivate some of these keys using "poke" commands, or you may wish to allow the student to redisplay lost material or to return to certain points in the program.

8. According to Kehrberg, students should always be given a status report at the end of the program so they will know how well they did (Kehrberg, 1979). In addition, consider ways of informing the student's teacher of his/her success. One problem with microcomputers without disk systems or printers is that it is difficult to record and save student data. One possibility for informing the teacher of the student's success is to freeze the display at the end of the program containing data about the student's performance. Another possibility is to use the speaker available on some microcomputers like the Apple to alert the teacher that the student has finished and that performance data will be displayed or that such a display can be obtained by entering a code that only the teacher knows. Another way to inform the teacher what the student has done is to have the student turn in paper on which computations have been done. You may even wish to include masters for student worksheets in the documentation that accompany the microcomputer software.

In order to make programming and debugging easy for yourself, make your program clear and readable by someone else--if they can read and understand the program, then you will be able to when bugs turn up and you must modify it months after you wrote it. Also, write your program so that the least experienced user will not be baffled by it.

Curriculum Considerations

Much of the software available for mathematics in the elementary grades deals with drill and practice in memorizing single-digit operations, and in multi-digit algorithms for whole numbers. A typical format is for the computer to display a problem on the screen, ask for the student to input the correct answer or to choose between several possible answers, then to inform the student if he/she is correct. If incorrect, the student is given another chance to give the correct answer.

This format may be somewhat motivational for students because they are working with a medium that is new to them and because they are informed immediately as to the correctness of their answer, but it certainly does not make full use of the great potential of the microcomputer. This type of drill might best be left for the traditional pencil and paper format while the microcomputer is used for what it does best.

Since so much time is spent in the typical elementary classroom teaching children computational algorithms and in drill and practice with them and with "basic facts" (single-digit operations), what the teacher really needs is software that teaches and practices skills and concepts that they have difficulty getting to but are extremely important to students' understanding of mathematics. Following are some suggestions for areas of the curriculum that need good software.

1. First and foremost is place value and understanding of the meaning of multi-digit numerals. Research (e.g., Engelhardt and Wiebe, 1981) has identified lack of understanding of these concepts to be a major reason for difficulties in mathematics at the upper elementary level. Software could develop these understandings first through pictorial models such as place-value charts, pictures of numeration blocks, etc. (See Figure 2), identifying place values in written numerals, expressing numerals in expanded notation, and so on.
2. As was mentioned, problem solving must be a major thrust of the elementary mathematics curriculum. The microcomputer seems to be especially well suited for instruction in this area.
3. Concepts of grouping and sets.
4. Geometry concepts.
5. Spatial visualization. This ability has been shown to correlate highly with success in abstract mathematics. It is possible that training with physical objects and pictorial representation on the microcomputer will improve students' abilities to visualize and mentally manipulate objects. An example would be to display pictures of "paper" that could be folded to form cubes with appropriate markings on certain sides. Students are then asked to mentally visualize how they could be folded together to form cubes and where the marked sides would be when the cube was formed. Further enhancement could involve the step by step folding of the cube, perhaps under direction of the student using the program.

Graphics

Graphics have already been discussed above. A few additional comments are in order, however. There are several major areas of use of microcomputer graphics. Three of these are to illustrate, to highlight, and to motivate.

Graphic illustrations can take numerous forms. One possibility is the use of pictorial models as discussed above. Another is to represent problem situations. Another is to represent real-life applications related to problems or computational exercises. In all of these uses, major benefits of the microcomputer over the textbook are displaying sequences of graphics that simulate movement or that picture sequences of events, allowing the user to control certain movements, and branching on the basis of user input. For example, in the subtraction example (Figure 2), each of the tens in Frame 2 could be labeled with a letter. The student could then be asked which of the rods should be removed. Another example would be to allow the student to control the number of rows and columns in an array representing different products.

Graphics can also be used to draw the student's attention to apropriate text or part of a pictorial representation. For example, suppose you are teaching problem solving skills and wish to draw the student's attention to extraneous information in the written problem. Reverse image , flashing words, arrows, boxes around the appropriate values could be used to highlight important words or numbers. Or, when using pictorial representations of mathematical operations or concepts, you could ask the child to point to a part of the picture to answer a question, then highlight the correct part of the design and ask the student if he/she pointed to the correct spot in the picture (e.g., in Figure 2 above you could ask the student to point to the "tens").

There are now may programs on the market that use graphics for motivational purposes. Light displays, cartoon figures that perform certain acts when the student correctly answers questions, faces that smile when a correct answer is recorded are all examples. In drill and practice programs, graphics can also be used to involve the student in a game or puzzle. For example, the display could contain a "racetrack" around the periphery of the screen. Then, when the student answers correctly the exercises or problems displayed in the center of the screen, her token is moved along the track. In drill on memorization of single-digit sums and products, time limits could be set by the teacher or by the student. Differing time limits and level of difficulty of problems would allow students of unequal ability to compete and have equal chances of winning.

Documentation

Documentation (user support material) is almost as important as the software itself. It should clearly give instructions as to how to load the program, how to run it, how to get out of trouble spots (e.g., how to correctly enter answers to questions). It should be written in the language of the novice user: if it is, probably the teacher will be able to understand it and be able to help the student out when trouble occurs.

Documentation should be tested along with the software to make sure that it does what it should. Do not be afraid to use pictorial helps. For example, show a picture of the correct way to load the cassette into the tape recorder, the correct buttons to push on the keyboard to activate the program, etc. Nothing is quite as frustrating to the user as instructions that do not make sense or that must be read six times to be understood.

Another aspect of documentation that is sometimes forgotten are specifications and instructions for the teacher or administrator who will be deciding when and where to use it. Do not laden this portion of the write-up with educational or computer jargon or acronyms. Remember that most users of your software will be relative novices and that those who are not will take what they need from your documentation and ignore the rest.

Testing and Debugging

Once your program is written, it should be thoroughly evaluated in classroom conditions. Wasserman (1979) differentiates between testing and debugging--testing is the attempt to determine if errors exist and debugging is the attempt to locate these, known errors.

Testing is first done by the person developing the software with an attempt at going through all possible branches in the program and attempting to anticipate all possible types of input. Kehrberg (1979) suggests newly created programs should be run with the deliberate intention of "bombing" it out. Enter unexpected input such as "return" without inputting any data, very long responses, and numeric data where string data is expected. He also suggests giving the program to malicious users (people who enjoy pointing out flaws in the work of others), and finally to naive users.

By observing the naive user, you will discover if your directions are well written, if he/she can run the program without the need to refer to external information (Heck, Johnson & Kansky, 1981), and if the software is appropriately designed. Your own testing coupled with that of others will help you detect systems errors (which should be avoided) and design considerations.

The computer has the possibility to meet individual needs in the mathematics classroom like no other tool. Within a few years, this tool will be available to almost all schools in the country. But, because it is so new, we are just beginning to find ways to put it to full use in the classroom. Research is needed in factors influencing its success in a variety of settings and with a variety of users. It is doubtful that the typical elementary or secondary teacher will have the background, ability or inclination to develop good software for use in their classrooms. What is needed are people with strong knowledge of microcomputers and with classroom experience and understanding of mathematics curriculum and learning factors to develop quality software for a large variety of uses in the classroom.

References

An agenda for action: Recommendations for school mathematics of the 1980s. Reston, Virginia: National Council of Teachers of Mathematics, 1980.

Caldwell, R. Guidelines for developing basic skills instructional materials for use with microcomputer technology. *Educational Technology*, 1980,

Engelhardt, J.M. & Wiebe, J.H. An investigation of the underlying assumption of error analysis. *Focus on Learning Problems in Mathematics*, 1981, 3(1), 13-24.

Heck, W.P., Johnson, J. & Kansky, R.J. *Guidelines for evaluating computerized instructional materials.* Reston, Virginia: National Council of Teachers of Mathematics, 1981.

Kantowski, M.G. Problem solving. In, Fennema, E. (Ed.), *Mathematics education research: Implications for the 1980s.* Alexandria, Virginia: ASCD, 1981.

Kehrberg, K.T. Microcomputer software development: new strategies for a new technology. *AEDS Journal*, 1979, 13(1), 103-110.

Moursund, D. *Teacher's guide to computers in the elementary school.* LeGrande, Oregon: International Council for Computers in Education, 1980.

Riedesel, C.A. & Burns, P.C. Research on teaching of elementary-school mathematics. In R.M. Travers (Ed.), *Second handbook of research on teaching.* Chicago: Rand McNally, 1973.

Suydam, M.N. & Higgins, J.L. *Activity-based learning in elementary school mathematics: Recommendations from research.* Colombus, Ohio: ERIC/SMEAC Science, Mathematics, and Environmental Education

Information Analysis Center, 1977.

Suydam, M.N. & Weaver, J.F. *Using research: A key to elementary school mathematics*. Colombus, Ohio: ERIC Information Analysis Center for Science, Mathematics, and Environmental Education, 1975.

Wasserman, A.I. On the meaning of discipline in software design and development. In Freeman, P. & Wasserman A.I. (Eds.), *Tutorial on software design techniques, second edition*, Long Beach, CA: IEEE Computer Society, 1977.

Software Arts
Jeffrey Raynor
Karen Raynor
3668 W. Aster
Phoenix, AZ 85029

EDUCATIONAL SOFTWARE DOESN'T HAVE TO BE BORING

Quality educational software is not as difficult to find as it once was since educators are beginning to demand programs which make full use of the computer's sophisticated capabilities. High resoulution graphics, animation and sound enhancement are an integral part of the latest quality software packages.

An early example of fine quality educational software is the Lemonade program from the Apple 3.2 system Master Diskette. This program, a simulation of the finances involved in the managing of a lemonade stand, combines colorful full-screen graphics, animation as well as familiar songs to keep the student interested, involved and learning.

Although there are now many fine commercial programs available, there is still a great need for high quality educational software. In the past, many of the fiascos in the educational software arena, were a result of the efforts of either programers who were not teachers or teachers who were not programmers. In both cases, the result was less than effective.

In the early stages of software development, programming the "fancies" so to speak, was tedious and difficult. Now, however with the advent of several good skeletal authoring systems, it takes very little programming background to create adequate programs. Add to this basic program, a sprinkling of Hi-resolution graphics, musical background, animation or a bit of 3-d graphics, and the "adequate" program becomes alive and effective.

HI-RESOLUTION COLOR GRAPHICS

There are currently many "paddle graphic" systems you can use to draw pictures on the screen, however the easiest method, by far, is to utilize a graphics tablet. one of the easiest to use and the most reliable is probably the original Apple Graphics Tablet which accepts direct surface drawing. The tablet's fine line discrimination makes creating pictures in this manner, both quick and enjoyable. At the present time the Apple

Graphics Tablet retails for about $725.

For the poor man's budget, there are other less sophisticated graphics tablets such as Versa Computing's Versawriter. The tablet board has a fixed arm which traces images from simple patterns placed below the mylar overlay. These shapes can then be stored as a complete pictures or stored as individual shapes in a shape table. The Versawriter has a good color selection and with a bit more time, can produce outstanding results. Using the newly created images is fast and simple. The Versawriter tablet and software lists for about $249.

One of the newest features in graphic tablet capabilities is the addition of functions which allow the user to trace objects in 2 or 3-d perspective. One such piece of equipment is the Space Tablet by Penguin Software which carries a list price of $395.

For those who cannot afford a graphics tablet, or simply prefer to create Hi-resolution images via the "paddle graphics" method, there are several excellent and easy to use programs currently available. With a little more work, these programs produce outstanding graphics. Some of these programs also provide portions to allow the user to fully or partially animate segments of the new creation. An exellent example is the E-Z Draw by Sirius Software which retails for $49.95. The E-Z Draw is a complete graphics editor which will also provide the user with 18 different lettering fonts in addition to the usual program aids such as draw, color, fill, rotate, enlarge and so forth.

Other fine programs which might also be useful are The Complete Graphics System by Penguin Software ($59.95) or Penguin's Special Effects enhancement ($39.95), the Super Shape Draw and Animate by Avant Garde Creations ($34.95), Higher Graphics II by Synergistic ($35.00), Bill Budge's 3-D Graphics System and Game Tool by California Pacific ($39.95), A2-GE1 Graphics Package by SubLogic ($59.95). The final choice of software would probably be determined by which special features the user is looking for.

For the user seeking a good all around programming aid, the Apple DOS Tool Kit ($69.00) is a good investment. The Tool Kit contains a Hi-res character generator, animation routines, assembler/editor as well as various utility routines called the Programmer's Assistant.

MUSIC AND SOUND

Adding sound to your programs can consist of anything from adding a few beeps or tones, to having the computer actually talk to the user. The easiest method is of course, to use the Apple's own built in speaker or even the programmer's resident music routine on Apples having Integer Basic resident in ROM.

There are also a few commercial programs available with pre-programmed sounds such as wails, sirens and the like which can be directly inserted into a new program.

Adding voice capabilities to an Apple can take many forms. On the simplist level, the Apple can be programmed to turn on and off a cassette tape recorder at appropriate times, thus providing a cassette-visual learning experience for the user. At the present time, several Educational software manufacturers have indeed chosen this format for some of their educational software materials.

On a slightly more sophisticated level, simple words and phrases can be inserted into programs from a source program without the use of additional hardware with a program such as The Voice by Muse. The speech is generated directly through the Apple speaker and although intelligible, would probably not be the best for daily instruction.

Speech generated by the addition of a voice synthesizer board has now become highly sophisticated thanks to advances in digitized speech. With an investment of $300 or less, the Apple around $300.00 the computer can speak to the user using it's own "voice" without sounding monotone and lifeless.

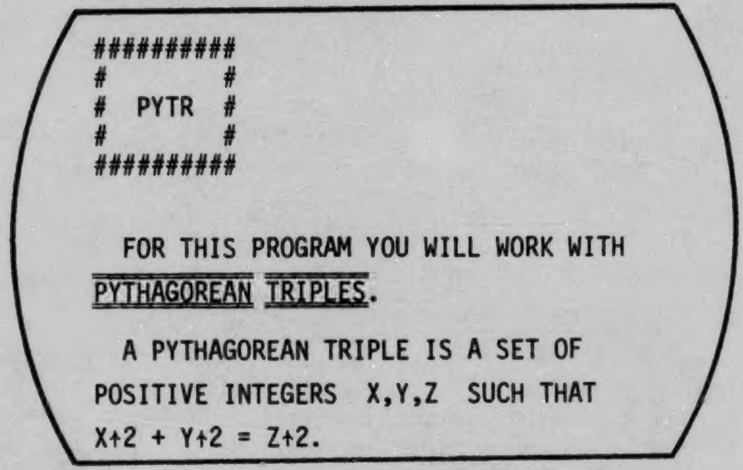
Although there are several excellent synthesizers currently on the market, One easy to use, highly intelligible voice synthesizer is Mountain Computing's Super Talker. The programmer speaks into a microphone and his voice is then digitized for later recall. The various words and phrases are stored as a phrase table, in much the same fashion that shapes are stored on a shape table. The programmer then recalls these words or phrases merely by selecting the sequence in which he wishes them to be spoken. Although some memory space is sacrificed for quality, the integrity of the spoken voice is such that it is easily identifiable. The addition of speech capabilities makes the computer seem much more personable and inviting.

As we have seen, there are now so many programming "aids" available, there is really little excuse for educational software to be lifeless and just plain BORING! As educators, we have long since recognized that classroom lessons have to be varied to be interesting and it's just about time we found out that software also needs these same qualities or it will soon lose its appeal even for the most die-hard computernik. Let's use those imaginations and create software with volcanoes which erupt in simulation, diagrams which actually explain themselves or experiments which patiently show the exact steps a student should follow through animated demonstration. When we start thinking in these terms when we preview or create our educational software, then we will have a dynamic instructional tool at our disposal.

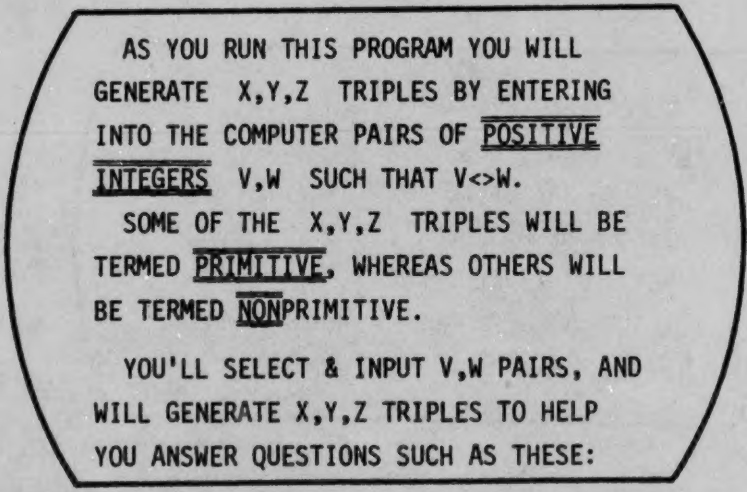
MATHEMATICAL PROBLEM SOLVING AND MICROCOMPUTERS: ONE WAY TO USE THEM JUDICIOUSLY

J. F. Weaver
Department of Curriculum and Instruction
The University of Wisconsin-Madison

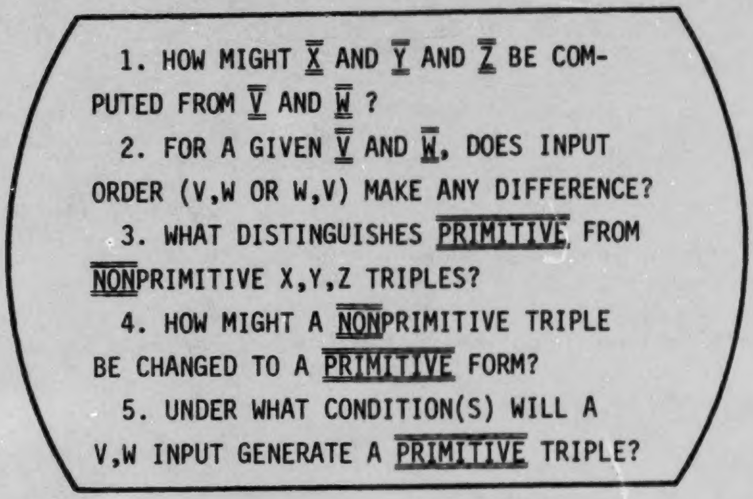
Imagine that you have loaded a program into a microcomputer, that you have given it a RUN instruction, and that you have followed several prompts that were displayed (but are not reproduced here), resulting in the following three things observed in succession on the CRT screen: (The first two displays are reproduced below; the third, at the top of the next page.)



```
##########
#        #
#  PYTR  #
#        #
##########


 FOR THIS PROGRAM YOU WILL WORK WITH
PYTHAGOREAN TRIPLES.

 A PYTHAGOREAN TRIPLE IS A SET OF
POSITIVE INTEGERS  X,Y,Z  SUCH THAT
X↑2 + Y↑2 = Z↑2.
```



```
 AS YOU RUN THIS PROGRAM YOU WILL
GENERATE  X,Y,Z  TRIPLES BY ENTERING
INTO THE COMPUTER PAIRS OF POSITIVE
INTEGERS  V,W  SUCH THAT V<>W.
 SOME OF THE  X,Y,Z  TRIPLES WILL BE
TERMED PRIMITIVE, WHEREAS OTHERS WILL
BE TERMED NONPRIMITIVE.

 YOU'LL SELECT & INPUT V,W PAIRS, AND
WILL GENERATE X,Y,Z TRIPLES TO HELP
YOU ANSWER QUESTIONS SUCH AS THESE:
```



1. HOW MIGHT X AND Y AND Z BE COMPUTED FROM V AND W ?
2. FOR A GIVEN V AND W, DOES INPUT ORDER (V,W OR W,V) MAKE ANY DIFFERENCE?
3. WHAT DISTINGUISHES PRIMITIVE FROM NONPRIMITIVE X,Y,Z TRIPLES?
4. HOW MIGHT A NONPRIMITIVE TRIPLE BE CHANGED TO A PRIMITIVE FORM?
5. UNDER WHAT CONDITION(S) WILL A V,W INPUT GENERATE A PRIMITIVE TRIPLE?

Imagine further that you, a student, . . .

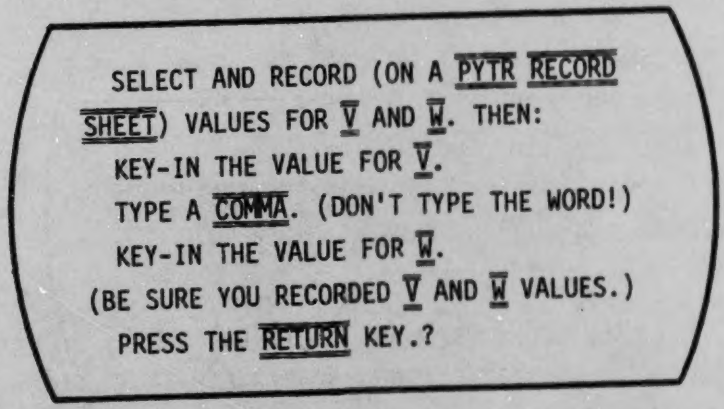
know in connection with the first display that the symbolic statement X↑2 + Y↑2 = Z↑2 is the computer's way of asserting that $X^2 + Y^2 = Z^2$;

know in the second display that V<>W is "computerese" for asserting that $V \neq W$ (V is not equal to W);

know what positive integers are;

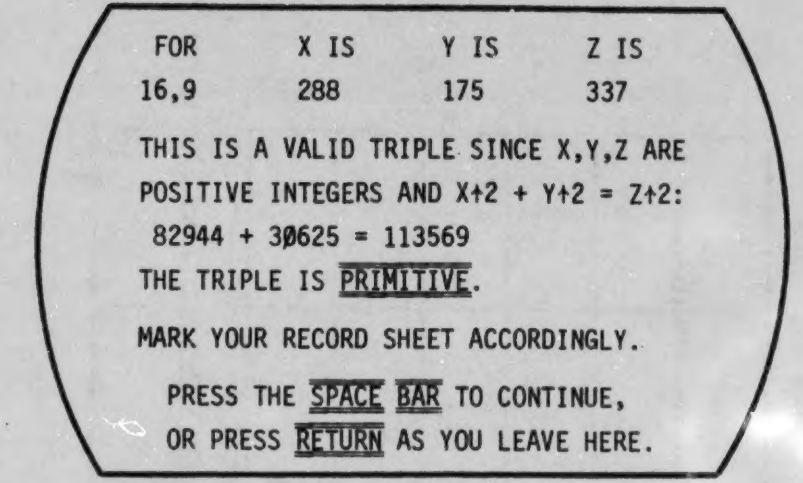
have been given a printed copy of the preceding three displays for future reference as may be needed, along with a printed PYTR Record Sheet form to be used subsequently (and subsequently reproduced in connection with this "paper").

Assume, now, that you followed the prompt below and tried these three V,W pairs in turn: 16,9 and 8,12 and 10,10:

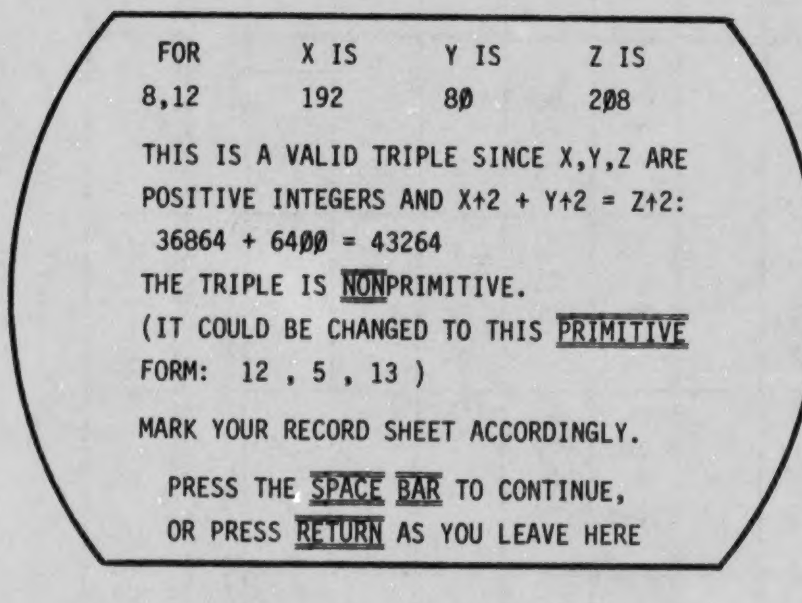


SELECT AND RECORD (ON A PYTR RECORD SHEET) VALUES FOR V AND W. THEN:
KEY-IN THE VALUE FOR V.
TYPE A COMMA. (DON'T TYPE THE WORD!)
KEY-IN THE VALUE FOR W.
(BE SURE YOU RECORDED V AND W VALUES.)
PRESS THE RETURN KEY.?

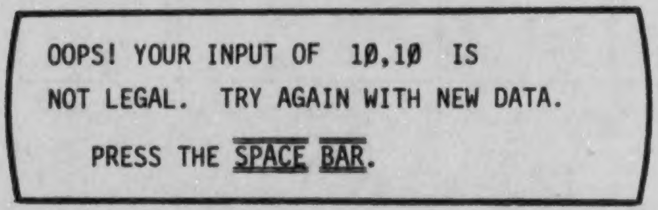
The resulting CRT displays appear on the following page, with a facsimile PYTR Record Sheet on the page after that.



```
 FOR        X IS      Y IS      Z IS
16,9        288       175       337

THIS IS A VALID TRIPLE SINCE X,Y,Z ARE
POSITIVE INTEGERS AND X↑2 + Y↑2 = Z↑2:
 82944 + 30625 = 113569
THE TRIPLE IS PRIMITIVE.

MARK YOUR RECORD SHEET ACCORDINGLY.

  PRESS THE SPACE BAR TO CONTINUE,
  OR PRESS RETURN AS YOU LEAVE HERE.
```



```
 FOR        X IS      Y IS      Z IS
8,12        192       80        208

THIS IS A VALID TRIPLE SINCE X,Y,Z ARE
POSITIVE INTEGERS AND X↑2 + Y↑2 = Z↑2:
 36864 + 6400 = 43264
THE TRIPLE IS NONPRIMITIVE.
(IT COULD BE CHANGED TO THIS PRIMITIVE
FORM:  12 , 5 , 13 )

MARK YOUR RECORD SHEET ACCORDINGLY.

  PRESS THE SPACE BAR TO CONTINUE,
  OR PRESS RETURN AS YOU LEAVE HERE
```



```
OOPS! YOUR INPUT OF  10,10  IS
NOT LEGAL.  TRY AGAIN WITH NEW DATA.

   PRESS THE SPACE BAR.
```

PYTR Record Sheet

Your input †		If your input generated a triple, record the values of X, Y and Z			If the triple is NONprimitive, record its PRIMITIVE form		
V	W	X	Y	Z	X	Y	Z
16	9	288	175	337			
8	12	192	80	208	12	5	13
10	10	Oops!					

† Although not mandatory, it is suggested that for V and W you select values that are less than 100.

(Oops: you apparently overlooked or forgot the input condition that V<>W.)

Be that as it may, I'll now relieve you from playing the role of a student actually using the program. We'll consider together the principal points that I wish to illustrate by this program example.

1. A person using PYTR as a problem solving situation does not have to be at or on the computer constantly during the course of the problem solving process. Cogitation, discussion with some other person(s), and possibly other non-computer activities, are desirable, vital parts of the problem solving process.

2. It is senseless and wasteful of precious time to have a person "glued" to a computer throughout a problem solving experience in which that computer is used somewhat intermittently, as illustrated by PYTR. It is wise to design the problem solving activity with its computer component in such a way that persons can readily alternate between being on/at the computer and off/away from it, as needed. When computer use is somewhat "fluid," as it were, precious time may be used more efficiently with a larger number of persons.

PYTR was but one of several exemplars used at this Conference session to illustrate different kinds and levels of mathematical content used as bases for problem solving activities in which computer use was intermittent, albeit not always of the same nature or for the same purpose. In each instance, however, the focus was upon "fluid" use of the computer as desirable,--not only from the standpoint of using time (at/on computers) more efficiently, but also from the standpoint of roles that computers may play in connection with the process of mathematical problem solving.

In connection with PYTR and the other exemplars, attention also was directed to the role(s) of the teacher in the problem solving process. The nature and extent of teacher "guidance" were not the same for every problem situation, nor would they be the same for every student.

The improvement of mathematical problem solving within the instructional context also is dependent upon the nature and quality of the supplementary or auxiliary print or similar materials used in conjunction with the computer. These, too, were illustrated for PYTR and the other exemplars during the course of this Conference session.

Persons having a further interest in PYTR per se will find additional information in Appendix A (see next page), along with the computer program in Appendix B.

APPENDIX A

Positive integers X,Y,Z constitute a pythagorean triple if

$$(1)\quad X^2 + Y^2 = Z^2$$

which is a second-degree diophantine equation.

Such triples may be generated from pairs of positive integers V,W for which $V > W$, using these or similar formulas:

$$(2)\ X = 2VW,$$

$$(3)\ Y = V^2 - W^2, \text{ and}$$

$$(4)\ Z = V^2 + W^2.$$

Note that if formula (3) is rewritten as

$$(3')\ Y = |V^2 - W^2|$$

it then is sufficient to impose the condition that $V \neq W$ (instead of the condition that $V > W$).

Let it be understood that HCF means highest common factor. Then:

A pythagorean triple is termed primitive if HCF(X,Y,Z) = 1. Otherwise, if HCF(X,Y,Z) ≠ 1, a triple is termed nonprimitive.

The preceding formulas -- (2), (3) or (3'), (4) -- generate a primitive pythagorean triple whenever V and W are relatively prime and of opposite parity (i.e., one odd, the other even).

Any nonprimitive triple X,Y,Z may be changed to a primitive form by dividing each member of X,Y,Z by HCF(X,Y,Z).

APPENDIX B

Note these things about the typescript of the following program, written in BASIC for a Commodore PET microcomputer:

? is used as an abbreviation for PRINT, except when it appears as part of text within double quotes.

Whenever the CLR HOME key is indicated, it is to be used with the SHIFT key also.

Characters with ═ above and below them are to be in "reverse field" format.

```
       CLR
100 ?"HOME
101 ?"#########
102 ?"#       #
103 ?"#  PYTR #
104 ?"#       #
105 ?"#########
107 ?:?:?
108 GOTO 375
```

```
       CLR
109 ?"
      HOME
110 ?"  FOR THIS PROGRAM YOU WILL WORK WITH
111 ?
112 ?"PYTHAGOREAN TRIPLES.
113 ?:?
114 ?"  A PYTHAGOREAN TRIPLE IS A SET OF
115 ?
116 ?"POSITIVE INTEGERS  X,Y,Z  SUCH THAT
117 ?
118 ?"X↑2 + Y↑2 = Z↑2 .
120 ?:?
121 ?"  FOR MORE INFORMATION,
122 ?
123 ?"  PRESS THE SPACE BAR.
125 GET C$:IF C$<>CHR$(32) GOTO 125
       CLR
130 ?"
      HOME
131 ?"  AS YOU RUN THIS PROGRAM YOU WILL
132 ?
133 ?"GENERATE  X,Y,Z  TRIPLES BY ENTERING
134 ?
135 ?"INTO THE COMPUTER PAIRS OF POSITIVE
136 ?
137 ?"INTEGERS  V,W  SUCH THAT V<>W.
138 ?
139 ?"  SOME OF THE  X,Y,Z  TRIPLES WILL BE
140 ?
141 ?"TERMED PRIMITIVE, WHEREAS OTHERS WILL
142 ?
143 ?"BE IDENTIFIED AS NONPRIMITIVE.
145 ?:?
146 ?"  YOU'LL SELECT & INPUT V,W PAIRS, AND
147 ?
148 ?"WILL GENERATE X,Y,Z TRIPLES TO HELP
149 ?
150 ?"YOU ANSWER QUESTIONS SUCH AS THESE:
151 ?:?
152 ?"  [PRESS THE SPACE BAR.]
155 GET C$:IF C$<>CHR$(32) GOTO 155
       CLR
160 ?"
      HOME
161 ?"  1. HOW MIGHT X AND Y AND Z BE
162 ?
163 ?"COMPUTED FROM V AND W ?"
164 ?
165 ?"  2. FOR A GIVEN V AND W, DOES INPUT
166 ?
167 ?"ORDER (V,W OR W,V) MAKE ANY DIFFERENCE?"
168 ?
169 ?"  3. WHAT DISTINGUISHES PRIMITIVE FROM
170 ?
171 ?"NONPRIMITIVE X,Y,Z TRIPLES?"
172 ?
```

```
173 ?"  4. HOW MIGHT A NONPRIMITIVE TRIPLE
174 ?
175 ?"BE CHANGED TO A PRIMITIVE FORM?"
176 ?
177 ?"  5. UNDER WHAT CONDITION(S) WILL A
178 ?
179 ?"V,W INPUT GENERATE A PRIMITIVE TRIPLE?"
180 ?:?
182 ?"  NOW, TO GET STARTED GATHERING DATA:
183 ?
184 ?"       PRESS THE SPACE BAR.
185 GET C$:IF C$<>CHR$(32) GOTO 185
189 ?"CLR HOME
190 ?"  SELECT AND RECORD (ON A PYTR RECORD
191 ?:?
193 ?"SHEET) VALUES FOR V AND W.  THEN:
195 ?:?
196 ?"  KEY-IN THE VALUE FOR V.
197 ?:?
199 ?"  TYPE A COMMA. (DON'T TYPE THE WORD!)
200 ?:?
202 ?"  KEY-IN THE VALUE FOR W.
203 ?:?
205 ?"(BE SURE YOU RECORDED V AND W VALUES.)
206 ?:?
207 ?"  PRESS THE RETURN KEY.";
208 INPUT V$,W$
209 V=VAL(V$):W=VAL(W$)
210 ?"CLR HOME
211 REM-REJECT INVALID INPUT
212 IF V< 0=OR W<=0 THEN 220
213 IF V<>INT(V) OR W<>INT(W) THEN 220
214 IF V  W THEN 230
220 ?"OOPS! YOUR INPUT OF  "V$","W$"  IS
221 ?
222 ?"NOT LEGAL.  TRY AGAIN WITH NEW DATA.
223 ?:?
225 ?"   PRESS THE SPACE BAR.
226 GET C$:IF C$<>CHR$(32) GOTO 226
227 GOTO 189
229 REM-COMPUTE X,Y,Z; VERIFY PYTHAG. PROPERTY
230 X=2*V*W:Y=ABS(V*V-W*W):Z=V*V+W*W
231 IF X*X+Y*Y=Z*Z THEN 237
232 ?:?"SOMETHING'S GONE HAYWIRE!
233 ?
234 ?"CHECK WITH YOUR INSTRUCTOR.
235 GOTO 399
236 REM-COMPUTE HCF(X,Y) FOR HCF(X,Y,Z)
237 P=X:Q=Y
238 R=P-Q*INT(P/Q)
239 IF R=0 THEN 241
240 P=Q:Q=R:GOTO 238
```

```
241 ?" FOR"," X IS"," Y IS", "Z IS
245 ?
250 ?V$","W$,X,Y,Z
260 ?:?
265 ?"THIS IS A VALID TRIPLE SINCE X,Y,Z ARE
270 ?
275 ?"POSITIVE INTEGERS AND X↑2 + Y↑2 = Z↑2:
280 ?
285 ?X*X+Y*Y=Z*Z
290 ?
294 REM-Q=HCF(X,Y,Z) FOR PRIM. VS. NONPRIM.
295 IF Q=1 THEN 315
300 ?"THE TRIPLE IS NONPRIMITIVE.
304 REM-CHANGE TRIPLE TO A PRIMITIVE FORM
305 X1=X/Q:Y1=Y/Q:Z1=Z/Q
306 ?
307 ?"(IT COULD BE CHANGED TO THIS PRIMITIVE
308 ?
309 ?"FORM: "X1","Y1","Z1")
310 GOTO 320
315 ?"THE TRIPLE IS PRIMITIVE.
320 ?:?
330 ?"MARK YOUR RECORD SHEET ACCORDINGLY.
335 ?:?
340 ?"  PRESS THE SPACE BAR TO CONTINUE,
345 ?
350 ?"  OR PRESS RETURN AS YOU LEAVE HERE.
355 GET C$
360 IF C$=CHR$(32) GOTO 189
361 IF C$=CHR$(13) GOTO 365
362 GOTO 355
       CLR
365 ?"
       HOME
370 FOR W=1 TO 2000:NEXT
375 ?"DO YOU WANT BACKGROUND INFORMATION?"
380 ?:?
385 ?"TYPE Y FOR YES OR N FOR NO.
390 GET C$
395 IF C$="Y" GOTO 109
396 IF C$="N" GOTO 189
397 GOTO 390
399 ?:?
400 END
```

Note: Since the preceding program is a revised version of a previous one, the line numbering is more "strange" or irregular than normally would be the case.

V. LITERACY

COMPUTER LITERACY AS CONTENT

Janis Rosenblum
Mathematics Teacher
Scottsdale School District
3811 N. 44th Street
Phoenix, Arizona 85018

Shirley Frye
Mathematics Coordinator
Scottsdale School District
3811 N. 44th Street
Phoenix, Arizona 85018

I. Introduction

  A. District's past program

  B. Plan for computer literacy

    1. Teachers
    2. Students

  C. Equipment

II. Scope and Sequence Content

  A. Goals and objectives

  B. Sample lessons

  C. Strategies

III. Teaching Ideas

  A. Course for teachers

  B. Alternatives for students

IV. Resources and Materials

V. Question - Answer

If airlines had progressed as rapidly in development as computers, the Concord would be carrying 500,000 passengers at 20,000,000 mph for less than 1¢ per person to London.

If the progress of manned flight had increased at the rate of computer technology, an astronaut could have orbited the earth nine years after the Kitty Hawk flight.

KNOWLEDGE	SKILLS
Development of Technology	Recognize that the computer came into existence because of its: -calculating speed -ability to control devices -ability to remember and recall data (data storage). Recognize the rapid development and increasing capabilities of technology.
The Hardware Components	Identify the five major components of a computer-input equipment, output equipment, memory unit, control unit, arithmetic/logic unit. Relate the components to the computers being used in the laboratory. Recognize how a computer works, including the input of data, processing of data, and output of data.
How a Computer Works	Understand the purpose of the stored-program. Recognize that the basic components of the computer perform as an interconnected system under the control of a stored program developed by a person.
The Language of Computers	Recognize that a computer needs instructions to operate and a special vocabulary. Recognize that a computer gets instructions from a program written in a programming language. Be aware there are several programming languages suitable for particular applications in business or science.

KNOWLEDGE	SKILLS
	Recognize the necessity of documentation (flowcharts, REM statements) in communicating with other programmers.
The Concept of an Algorithm (Procedure)	Follow and give the correct output for a simple algorithm using a flowchart.
	Construct a BASIC-language program which matches a given elementary numerical algorithm.
	Follow and give the correct output for an algorithm that includes a decision.
	Construct a BASIC-language program which corresponds to a given numerical algorithm with one loop involved.
Title - Modifying an Algorithm	Complete partially formulated algorithms in order to perform a specified function.
	Modify a given algorithm to produce a new, but related result.
	Expand an algorithm to produce a new, but related result.
Practical Uses of Computers	Identify three major categories of computer applications.
	Identify the most appropriate category of computer application for solving given problems.
	Estimate whether or not a given problem would be a practical application for a computer.

KNOWLEDGE	SKILLS
	Name four ways a computer can serve as a personal tool.
	Identify three major categories of computer applications and offer examples of each.
	Identify several computer applications that apply to education.
Things Computers Can and Cannot Do	Recognize that computers are generally good at information processing tasks that require speed, accuracy, data retrieval, and automatic operations.
	Recognize that appropriate use of computers is based on particular applications. (Different sizes or configurations of a computer will help determine its uses.)
	Recognize that some considerations for using computers are cost, software availability, and storage capacity.
	Describe some actual examples of how computers have been useful or detrimental to man.

COMPUTER REFERENCES

Cross, Wilbur, A Job With A future in Computers
Grosset and Dunlap, New York 1969

Dwyer, Thomas A. and Michael S. Kaufman, A Guided Tour of Computer Programming in BASIC
Houghton Mifflin Co., Boston 1977

Kohn, Bernice, Computers at Your Service
Prentice-Hall Inc., Englewood Cliffs, N.J. 1962

Rice, Jean, Teacher's Guide and Activity Book to Accompany "My Friend the Computer," T.S.
Denison and Company, Inc., Minneapolis, Minnesota 1976

Rusch, Richard B., Computers: Their History and How They Work
Simon and Schuster, New York 1969

Thomas, Shirley, Computers - Their History, Present Applications and Future
Holt, Rinehart and Winston, Inc., New York 1965

Treuenfels, Peter, Looking Forward to a Career in Computers
Dillon Press, Inc., Minneapolis, Minnesota 1975

BASIC Computer Language - Radio Shack, 1978

"Careers in Computer Science and Service" (slide presentation)
The Center for Humanities, Inc., Mount Krisco, New York 1982

"BASIC: An Introduction to Computer Programming" (filmstrip presentation)
Guidance Associates, Inc., Mount Krisco, New York 1981

"Time: From Moons to Microseconds" (filmstrip presentation)
Guidance Associates, Inc., Mount Krisco, New York 1975

COMPUTER LITERACY PROJECT

Bruce Eldredge
Prescott Unified School District No. 1
P.O. Box 1231
Prescott, Arizona 86302

1981 - 1982
SCHOOL YEAR

COMPUTER LITERACY PROJECT

Computers are not only here to stay, but they will become an integral part of our lives. Joseph Garber, a New York management consultant, predicts that, "Within ten or fifteen years, some degree of computer efficiency will be necessary in order for people to be employable." Beyond those whose jobs are directly affected by computers are thousands of other jobs that indirectly rely upon some data processing. The communications industry and the financial community, which play such important parts in our lives, are dependent upon computers. The role of educating the public about computers careers, social implications and the capabilities of computer falls to the schools.

BACKGROUND

In 1979 the Prescott Unified School District initiated a computer literacy project in grades four through twelve. The initial funding for the computers themselves and curriculum development came from ESEA Title IV-C and the State Vocational Education Program. The school district has also provided funding to purchase additional equipment. The four areas of computer literacy - programming, social impact, career exploration and the history of computers - involve all fourth through seventh graders. Additionally, elective courses for upper grade students in junior high and high school are offered.

We are now in our third year of the project. The original project grew from one teacher who brought his own machine into his class. Interest and motivation were high, so steps were taken to develop a more sophisticated project. The project began with thirty-two computers and reached the one hundred mark this year. We are also now beginning to place printers and disc storage devices in some applications.

ELEMENTARY

Serving our five elementary schools, we have two sets of twelve computers. These are rotated among the schools on a quarterly basis. Early in the project we used a ratio of three students per computer. We have since found that two students per computer is far more workable.

The four week literacy program spends one week before the scheduled arrival of the computers dealing with the social implications and history of computers. Some career exploration is also discussed. Teacher developed materials serve careers and social implications, and Marion Ball's text, What is a Computer? discusses the history.

The other three weeks deal with a sequential curriculum for each intermediate grade. These teacher developed materials guide the students through actual programming activities. Private and parochial schools in our area are offered the use of our computers throughout the year as the schedule permits.

JUNIOR HIGH

Our two junior high schools offer varying programs. One school schedules all seventh graders for a semester course in computers. This complements a semester course in science. They then have the option of an elective advanced course in eighth and ninth grades.

The other junior high school offers a six week mini-course in computer science for seventh graders. These students also have the option of a semester elective following completion of the required mini-course. Both schools also make the computers available to gifted students on a more individual basis.

HIGH SCHOOL

Prescott High School makes two courses in computer science available. Students may take the beginning course, or they may take the advanced course where we keep the student to computer ratio one to one. Those who show exceptional aptitude may continue their computer education through our local community college while still enrolled in high school.

TEACHER TRAINING

Perhaps the most critical area to the success of our project is our teacher training program. A few of the teachers have backgrounds in computers; however, most have been trained through a course offered in cooperation with our community college. "Computers in Education" is designed primarily for those teachers of grades four through twelve. The course follows the general outline of our elementary computer literacy curriculum for the first half of the course. The second half is devoted to development of programs useful in individual teaching situations. Approximately half of our present faculty has completed the course and can be considered computer literate. An advanced course has been offered as interest is expressed. Several teachers have also made a personal investment in computers for their own homes.

BROADENING THE BASE

Because of our teacher training, we are now seeing an increase in the use of computers outside the computer literacy program itself. Several math classes are using computers. Resource teachers at each of our five elementary schools use microcomputers to help their students with a variety of learning problems. This year one of the junior high librarians will be developing programs to handle circulation and other management. An English teacher is experimenting with word processing applications. A computer is being used to help manage a counseling department. Gifted students regularly have access to computers. A basic skills lab is in the planning stages.

CONCLUSION

Of course, predictions are not bound to happen, but we feel it is safe to say that computers will be an important part of the world our children inherit. By making computer literacy an important aspect of school life, hopefully we will be meeting the students' needs for their society and jobs in the 1980's.

PERFORMANCE OBJECTIVE(S): (Who will perform? What will be measured? What level of performance will be accepted? What was expected to happen?)

4th Grade

Programming

Loading and Saving Procedures
Using the computer as a calculator
Basic commands:
  VERIFY, LIST, NEW
Programming: GOTO, END
Beginning computer graphics

Careers

Awareness of the different types of careers available in the data processing field

History

Four generations of computers

Social Aspects

How computers may affect our lives in the future

5th Grade

Programming

Review 4th grade objectives
Introduce alphanumeric variables
Simple problem solving using the computer

Careers

How computers affect the student's personal life

History

Main components of a computer system

Social Aspects

Role playing activities dealing with misuse of computers

6th Grade

Programming

Review 4th and 5th grade curriculum
If - then statements
For - next loops
Advanced problem solving using the computer

Careers

Exploring careers that use computers as part of their daily routines

History

The development of the microprocessor (chip) and its significance to the present day computer

Social Aspects

Using a current events approach to discuss social ramifications of current uses of computers in our society

Junior High

Mini-course

Programming

- Review 4-6th grade curriculum
- Introduce Random number generators
- Cursor control keys for editing
- Write a simple computer game

Careers

Explore careers in data processing

History

In depth study of the history of computers

Social Aspects

Current events approach and role playing dealing with computer crime

Elective course

Programming

- Review basic statements and commands already introduced
- Introduce array variables
- Random number generator
- Screen graphics using poke statements

Independent Study course

PROJECT - Develop and market a computer game or graphic demonstration

- Students will write a game or graphic demonstration
- Students will explore potential markets for their program
- Students will contact these markets and submit their program on speculation to purchase
- Students will write a report summarizing the results of their project

High School

Introduction to Computers/Programming

Main Objective:

To introduce computer programming through the computer language of BASIC

Skills such as programming, flowcharting, documentation and problem solving are emphasized

(High School continued on next page...)

Topics covered during 18 week course are:
Calculator mode, screen editing, screen graphics, variable names, storing and retrieving programs
Also covered are uses of the following BASIC statements: REM, PRINT, LET, GOTO, IF-THEN, READ/DATA, FOR/NEXT, DIM, INPUT, AN and END

Advanced Computer Programming

This course is the follow-up 18 week course to "Introduction to Computers/Programming". It is recommended that the "Intro" course be taken before "Advanced" course is attempted

Main Objective:
To emphasize "programmer skills". Unlike the "Intro" course which emphasizes the language of BASIC, this course is designed to teach those skills which make up a good programmer, using BASIC as the programming language. Greatest emphasis will be placed on flowcharting, documenting and writing "readable" programs

Other topics covered:
Besides working on programming skills, there will be several topics related to extended BASIC, also known as X-BASIC. This includes use of tables, arrays, sorting, use of the various peripheral devices (printer, disk drives), special graphics, system subroutines, machine language and use of machine memory (PEEK/POKE)

Summary of Objectives

Mastery of performance objects will be determined by the following criteria:

Significant difference in Pre/Post tests
Instructor evaluation of final projects
Outside evaluation of over-all project

THE IMPORTANCE OF COMPUTERS TO EDUCATION

Dan Isaacson, Ph.D.
Publisher and Editor, CourseWare Magazine Programs;
Lecturer, Department of Information Systems
and Decision Sciences
California State University
Fresno, CA 93740
(209) 294-2823

> In less than 25 years, industry has witnessed the transformation of the computer from a behemoth weighing tons and requiring extensive human and physical resources, to a device about the size of an attache case. In fact, the evolution of computing has been similar to that of the printed book, but in 40 years rather than 600.
>
> A. C. Kay in
> Scientific American.

The printing press and the automobile

When the printing press was invented, education consisted of pure, rote learning. The teacher would dictate to the class while students memorized and regurgitated the dictated material. Visionary educators saw a change coming to education. Critics said books would never enter the classroom: they were too expensive, there would never be enough of them and reading was too difficult a skill.

At a later time, critics said the automobile would never be used except in cities because they were too expensive and there were no roads outside of cities. Today's critics are saying computers will never be used extensively in schools because they're too expensive and it is too difficult to learn to use them.

According to Infoworld News, in only six years, revenues from sales of microcomputing equipment have gone from nothing to almost two billion dollars. In Minnesota, M.E.C.C. (Minnesota Educational Computing Consortium) signed a 1978 state contract with the Apple Computer Company to supply Minnesota schools with Apple II computers at a special price. Within two years, 2000 Apple computers were bought by Minnesota school districts and M.E.C.C. is regarded as the foremost educational authority on microcomputing in schools. They have recently accepted a bid by the Atari Company which will put the power of an Atari 400 computer with one disk drive in the hands of their students for less than $500. Last year this amount of computing power cost over $1200. The state of Oregon is forming an Oregon Computing Consortium this year; they will be looking for similar advantages, and Dr. David Moursund, editor of The Computing Teacher Journal, has written that already there are over 100,000 personal computers in U. S. schools.

Our astronauts didn't get to the moon by saying it couldn't be done. Mankind walked on the moon because some visionaries said it could be done. There are other visions--kid's visions. Kids grow up using calculators and computer as naturally as we elders have used electricity, automobiles and television. We wonder how anyone could have lived before electricity, automobiles and television.

The electronic calculator has been thrust upon our classrooms by kids, not by teachers. In the six years from 1972 to 1978, the price of a four-function, hand calculator dropped from $179.95 to $5.95 (Sear's catalogs) and over 80 million calculators have been sold since 1978! In the next decade, kids will force us to accept the use of personal computers in our classrooms. Personal computer prices have fallen from $2500 in 1975 to $299 in 1981, and are still falling.

How does the computer differ from the teaching machine?

Teaching machine salesmen told us there would soon be a teaching machine in every classroom--it hasn't happened. Film projector salesmen told us there would soon be a film projector in every classroom--it hasn't happened. Computer salesmen tell us there will soon be at least one computer in every classroom--why should that happen any more than with the teaching machine or the film projector?

The State Department of Education of the State of Florida has recently signed a contract with CourseWare Magazine Programs to purchase the reproduction rights of CourseWare's programs and documentation for its 67 statewide school districts and 26 community college districts. The Oregon Computing Consortium is discussing a similar arrangement with CourseWare. M.E.C.C. has contracts with Apple and Atari. There are 100,000 computers in schools already in the United States and those have been installed at relatively high prices.

Obviously something about computers is different from the teaching machine and the film projector.

Several years ago, when I started writing educational programs for microcomputers, I tested out my first program on a third grade youngster. She sat in front of the computer doing addition facts drill problems. She sat there for 20 minutes working one problem after another. When she decided to stop, the computer reported she had completed 213 addition facts drill problems! Have you been able to match that record with flash card drill or paper and pencil worksheets?

A computer in the classroom is an educational pin-ball machine. If you've ever watched a youngster, enthralled with a pin-ball machine, spending hours on a non-educationally oriented skill, you can appreciate the excitement I find in providing youngsters with a highly motivational device which teaches too!

Outlook for the year 2000

In the year 2000, I see a computer in every home attached to the home television set. A.T. & T. plans to replace all its current analog telephone transmission lines with digital lines within the next decade. This revision will allow home computers to talk to other home computers or to business or educational computers without the current speed restrictions due to analog transmission. The United States Post Office service will have to change drastically as correspondence will travel directly from computer to computer in fractions of a second. Pilot projects are now taking place in Britain, France, and the U.S. with home computers capable of directly contacting an encyclopedic library of information and services.

With a computer in every home, I see kids learning and asking questions when questions occur, not solely during social studies or mathematics class periods. It has been said kids stop learning at 8:00 when they reach school and start learning again at 3:00 when school lets out! Are we wasting 12 years of our young people's lives? Many school subjects are no longer relevant to today's kids. Why do we still drill long division with 3 digit divisors in 6th grade when no one (yes, no one) ever does it by hand any more. A year of formal geometry seems to me to be irrelevant to today's youngsters. There are better ways to approach formal logic than through old man Euclid. I urge revising the curriculum to include more relevant and exciting training in problem solving skills and omitting 6th grade long division drill and 10th grade formal geometry.

The importance of computers to education

What should one take away from this meeting on the importance of computers to education? I believe we must recognize computers are here to stay. I believe we must recognize computers are powerful friends and use them accordingly. I believe we must recognize that kids will force us to change what and how we teach just because computers exist. Are we educators going to be dragged--kicking and screaming--dragged into the future by our students, or do we have the foresight to prepare ourselves for the "roadways that will be constructed outside our cities" and plan for the future rather than allow chaotic growth to inundate us.

References

Film - "Now The Chips Are Down." B.B.C. Films. Washington University Film Library (about $27 for three days rental).

The Computing Teacher Journal. Department of Computer and Information Science, University of Oregon, Eugene, OR 97403.

VI. PROGRAMMING

BASIC IN TEN EASY STEPS

by Marilyn Sue Ford
2234 Pommel Avenue
Las Vegas, Nevada 89119

STEP I - PRINT

The PRINT statement tells the computer to display information for the user.

•Printing strings (words)

```
10 PRINT "COMPUTERS ARE FUN."      RETURN*

RUN

  COMPUTERS ARE FUN.
```

•Printing problem solutions

```
20 PRINT 20/5

30 PRINT 3*46

LIST
10 PRINT "COMPUTERS ARE FUN."
20 PRINT 20/5
30 PRINT 3*46
RUN
  COMPUTERS ARE FUN.
  4
  138
```

•Printing number sentences

```
40 PRINT "12-7"

LIST
10 PRINT "COMPUTERS ARE FUN."
20 PRINT 20/5
30 PRINT 3*46
40 PRINT "12-7"
```

*Press the RETURN (or ENTER) key after typing a line or command.

```
RUN
  COMPUTERS ARE FUN.
  4
  138
  12-7
```

•Printing blank lines

```
25 PRINT
```

```
LIST
10 PRINT "COMPUTERS ARE FUN."
20 PRINT 20/5
25 PRINT
30 PRINT 3*46
40 PRINT "12-7"
RUN
  COMPUTERS ARE FUN.
  4

  138
  12-7
```

STEP II - PRINT

Given a program with the PRINT statement, determine the output. Then type the program into the computer and RUN it to check yourself.

Here's the program.

```
NEW
1C REM PRINT STATEMENT
20 PRINT "BIG"
30 PRINT
40 PRINT "YELLOW"
50 PRINT 5*7
60 PRINT "6-3"
70 END
```

Show the output.

```
RUN
  BIG

  YELLOW
  35
  6-3
```

STEP III - LET

The LET statement refers to the content of a memory location in the computer.

•Storing numeric data

```
10 LET N = 25

20 LET E = 2*N

30 PRINT N;E
```

```
RUN
  25  50
```

•Storing alphanumeric data

```
30 LET C$ = "I LIKE COMPUTERS."

40 PRINT N;E,C$
```

```
LIST
10 LET N = 25
20 LET E = 2*N
30 LET C$ = "I LIKE COMPUTERS."
40 PRINT N;E,C$
RUN
  25  50               I LIKE COMPUTERS.
```

STEP IV - LET

Given a program with the LET statement, determine the output. Then type the program into the computer and RUN it to check yourself.

Here's the program.

```
NEW
10 REM LET STATEMENT
20 LET Q$ = "I'M SPEEDY"
30 LET D = 36
40 LET P = 22
50 LET F = D*P
60 PRINT Q$
70 PRINT D;" x ";P;" = ";F
80 END
```

Show the output.

```
RUN
  I'M SPEEDY
  36 x 22 = 792
```

STEP V - INPUT

The INPUT statement accepts new data typed in by the user each time the program is RUN.

•Storing one value

```
10 PRINT "ENTER A NUMBER."

20 INPUT N
```

```
RUN
  ENTER A NUMBER.
  ? 16
```

•Storing more than one value

```
30 PRINT "ENTER 3 NUMBERS SEPARATED WITH COMMAS."

40 INPUT X,Y,Z
```

```
LIST
10 PRINT "ENTER A NUMBER."
20 INPUT N
30 PRINT "ENTER 3 NUMBERS SEPARATED WITH COMMAS."
40 INPUT X,Y,Z
RUN
  ENTER A NUMBER.
  ? 31
  ENTER 3 NUMBERS SEPARATED WITH COMMAS.
  ? 22, 34, 41
```

•Storing words and phrases

```
50 PRINT "ENTER YOUR NAME."

60 INPUT N$
```

```
LIST
10 PRINT "ENTER A NUMBER."
20 INPUT N
30 PRINT "ENTER 3 NUMBERS SEPARATED WITH COMMAS."
40 INPUT X,Y,Z
50 PRINT "ENTER YOUR NAME."
60 INPUT N$
RUN
  ENTER A NUMBER.
  ? 41
  ENTER 3 NUMBERS SEPARATED WITH COMMAS.
  ? 910, 17, 78
  ENTER YOUR NAME.
  ? MARILYN
```

STEP VI - INPUT

Given a program with the INPUT statement, determine the output. Then type the program into the computer and RUN it to check yourself.

Here's the program.

```
NEW
10 REM INPUT STATEMENT
20 PRINT "NAME A ROOM IN YOUR HOUSE."
30 INPUT R$
40 PRINT "ENTER THE LENGTH AND WIDTH SEPARATED WI
TH COMMAS."
50 INPUT L,W
60 PRINT "AREA OF ";R$
70 PRINT "IS ";L*W
80 END
```

Show the output.

```
RUN
  NAME A ROOM IN YOUR HOUSE.
  ? LIVING ROOM
  ENTER THE LENGTH AND WIDTH SEPARATED WITH COMMAS.
  ? 20, 14
  AREA OF LIVING ROOM
  IS 280
```

STEP VII - READ/DATA

The READ and DATA statements allow the programmer to store information the program will use.

- Storing one value

```
10 READ A

20 DATA 12

30 PRINT A
```

```
RUN
  12
```

- Storing more than one value

```
40 READ C,D

50 DATA 15, 17.5

60 PRINT C,D
```

```
LIST
10 READ A
20 DATA 12
30 PRINT A
40 READ C,D
50 DATA 15, 17.5
60 PRINT C,D
RUN
  12
  15             17.5
```

- Storing words or phrases

```
70 READ Y$,Z$

80 DATA  MARILYN, " LOVES COMPUTERS."

90 PRINT Y$;Z$
```

```
LIST
10 READ A
20 DATA 12
30 PRINT A
40 READ C,D
50 DATA 15, 17.5
```

```
60 PRINT C,D
70 READ Y$,Z$
80 DATA MARILYN," LOVES COMPUTERS."
90 PRINT Y$;Z$
RUN
  12
  15               17.5
  MARILYN LOVES COMPUTERS.
```

STEP VIII - READ/DATA

Given a program with the READ and DATA statements, determine the output. Then type the program into the computer and RUN it to check yourself.

Here's the program.

```
NEW
10 REM READ/DATA STATEMENTS
20 READ L$,L,W$,W
30 DATA LENGTH, 14, WIDTH, 12
40 PRINT "AREA OF ROOM ";L;" by ";W
50 PRINT "IS ";L*W
60 END
```

Show the output.

```
RUN
  AREA OF ROOM 14 by 12
  IS 168
```

STEP IX - PROBLEM

A bicycle wheel is 16 inches in diameter. How far will the bike travel during one revolution of the wheel?

Hint: Apply the circumference of a circle formula.
$C = \pi D$

Using only the PRINT statement, write a computer program to solve this problem.

Sample program.

```
10 PRINT 3.14 * 16
20 END

RUN
  50.24
```

STEP X - BICYCLE PROGRAM

Using the LET statement, write a computer program to solve the bicycle problem.

Sample program.

```
10 LET P = 3.14
20 LET D = 16
30 PRINT P*D
40 END

RUN
  50.24
```

Using the INPUT statement, write a computer program to solve the bicycle problem.

Sample program.

```
10 PRINT "WHEEL DIAMETER"
20 INPUT D
30 LET P = 3.14
40 PRINT P*D
50 END

RUN
  WHEEL DIAMETER
  ? 16
  50.24
```

Using the READ/DATA statements, write a computer program to solve the bicycle problem.

Sample program.

```
10 READ P,D
20 DATA 3.14, 16
30 PRINT P*D
40 END

RUN
  50.24
```

Sidenotes: Start each program with the command NEW. This clears the computer's memory.

Each microcomputer has a command to clear the screen. For example, on the TRS-80 type CLS to clear the screen.

Approximately 80% of the preceding programming outline was presented at a hands-on microcomputer workshop. Two to three participants crowded around eight TRS-80 Model IIIs while the remainder of the audience discussed the computer input and output for each of the BASIC programming steps. In the hour and a half workshop the novice programming participants covered all but steps 7 and 8. At the conclusion of the workshop, several participants claimed excitement over their acquisition of some BASIC programming skills --- enough knowledge to comfortably read and understand a beginning programming textbook or microcomputer programming manual.

Unfortunately, the outline omits much of the narrative explanation followed by the participant's questions with answers. Even so, any novice programmer with access to a microcomputer should be able to work through the programming steps. Just find the "on" switch and begin typing.

DON'T LET YOUR PROGRAMS BUG YOU

Dr. Homer Baker
Phoenix Union High School District
2526 West Osborn Road
Phoenix, Arizona 85017

Raymond Preston
1009 West Riviera Circle
Tempe, Arizona 85282

INTRODUCTION

Often poor programing techniques introduces errors in a program. Correcting those errors is made more difficult by the same poor planning. As a result, many programs take a great deal more time to program and debug than is necessary. Good programing techniques will reduce the time it takes to write and debug computer programs. The programing techniques introduced here are those we have found useful in producing programs.

Larger and more complex programs are more difficult because of the amount of information the programer needs to remember. We will provide two techniques which will help you organize and use this information. The first technique is blocking the program into logical modules. The second is the identification of variable names.

A benefit of using these techniques is that it will be easier to write and modify programs. The same programing techniques are also beneficial if the program is so large that you have to assign several people to the same program. The benefits will result in programs that:

a. have fewer errors
b. are easier to modify
c. are easier to correct.

This workshop will demonstrate some of the techniques for blocking programs and for the identification of variables. As we work through a program we will use those techniques to have several people writing the same program.

Our goal for this workshop is that you will be better equiped as a programer and this, in turn, will assist you in using a computer. We are going to spend the next 90 minutes putting a program together that will assist us in assigning grades to students. Because of the limitations of time we may not complete all of our program.

Some of the assumptions we have made about the participants in this workshop are:

1. We are all educators and we have an interest in using microcomputers in education.

2. We are looking for ways to improve our skills as educators.

3. There will be participants which have no experience with computers and, obviously, no experience with programing. Therefore, we will team the participants without experience with participants with experience.

GRADES PROGRAM

The specifications of the Grades program we will use in the workshop is in your handout. Check the specifications for information about the program. You will also find some computer definitions which may be useful to you.

Obviously, today we are not going to be able to test our program. Therefore, we have provided samples of the Grade program written in BASIC, FORTRAN, Pascal and COBOL. However, we hope that you refer to them as little as possible.

PROGRAMING TECHNIQUES

BLOCKING PROGRAMS

Principle 1: Divide and Conquer.

Subdivide program into logical blocks and then write and test each section individually. One of the problems with longer programs is their complexity makes it difficult to remember all of the parts and their interaction. However, each block is less complex and easier to understand and program. The first thing we want to do with the Grades program is to write the menu part of the program. You will do this as a cooperative team effort. As you work on this part of the program, select the team member who will become the team captian.

Guidelines to follow when blocking programs.

1. Write the program so that the flow of the program goes from top to the bottom. Use as few GO TOs as possible.

2. Make sure that each block does only one thing

well. Otherwise, each block will be as confusing as if it were a complete program.

3. Be specific about global variables. Others working with the program do not need to be surprised by false assumptions about what the variables were supposed to do.

4. Write and test each block separatly.

5. Write program specifications so that you would understand them next year.

SPECIFICATIONS

Specifications are a lot like flowcharts--no one does them because they take so much time and once they are done you don't need them....until something doesn't work like you think it should.

Specifications are useful when constructing large programs. Often you can identify and solve problems before they become troublesome. They are usually necessary when debugging or modifying large programs. For those of us who are ordinary programers, the specifications are useful a year later when we need to change a large program and we have forgotten how we programed part of the program.

Specifications are necessary when you have more than one person working on the same program. Programers who are working on separate sections need to know the names of the variables and routines so that proper communication can be established between parts of the program.

GRADES PROGRAM SPECIFICATIONS

The blocks of the Grades program are the options presented to the user. The options you will provide are: Quit, Add, Print, Delete, Change, Mean, Sort and Histogram.

Exercise #1

The first task is to write the menu portion of the program. We have included below a framework, in each of the target languages, to assist in developing the menu.

Include the Quit option as you program the menu portion of the program. This will allow you to get out of the program as you test each part. Your handout contains the program specifications.

Two variables are declared as global variables: the array where the scores are stored and the integer variable which contains the number of scores stored in the array.

Blocks and Global Variables in the BASIC Program

```
90 ***** MAIN PROGRAM
100 DIM A(100)
110 N=0
400 REM ***** ADD NEW DATA
600 REM ***** CHANGE A DATA VALUE
800 REM ***** DELETE A DATA ELEMENT
1000 REM ***** PRINT DATA
1200 REM ***** SORT DATA ARRAY
1400 REM ***** PRINT A HISTOGRAM OF THE SORTED DATA
1600 REM ***** PRINT MEAN OF DATA
1800 REM ***** QUIT THE PROGRAM
1810 END
```

Blocks and Global Variables in the Pascal Program

```
program grades;
var current_option:char;
    number_of_scores:integer;
    list_of_scores:array[1..100] of integer;
     procedure add;
     procedure change;
     procedure delete;
     procedure sort;
     procedure histogram;
     procedure mean;
     procedure print;
  begin {grades}
  end {grades}.
```

Blocks and Global Variables in the FORTRAN Program

```
       program grades
        integer list(100),noscor
        common list,noscor
       subroutine add
       subroutine change
       subroutine delete
       subroutine mean
       subroutine print
       subroutine sort
       subroutine histog
```

Blocks and Global Variables in the COBOL Program

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID.  GRADES.
    ENVIRONMENT DIVISION.
    CONFIGURATION SECTION.
    SOURCE-COMPUTER.  YOUR-COMPUTER.
    OBJECT-COMPUTER.  YOUR-COMPUTER.

    DATA DIVISION.
    WORKING-SECTION SECTION.
    01  NUMBER-OF-GRADES         PIC 999 VALUE ZERO.
    01  TABLE.
        02  TEMP-SCORE           PIC 999.
        02  LIST-OF-SCORES
            OCCURS 100 TIMES     PIC 999.

    PROCEDURE DIVISION.
    MENU.
    ADD-PARA.
    CHANGE-PARA.
    DELETE-PARA.
    HISTOGRAM-PARA.
    MEAN-PARA.
    PRINT-PARA.
    SORT-PARA.
```

TEAM PROGRAMING

Principle 2:

A second advantage of blocking programs is that different people can work on different blocks. The important thing to remember is that global variables must be maintained accurately by each block or the other blocks will not work correctly. For example, the scores will be kept in an array and it is important that all of the seperate blocks use the same array. Also, the number of scores is used by the different blocks in the program so it is important that it be kept accurately by each block.

Exercise #2

Select a team leader.

Subdivide the blocks amoung the people in your group. Assign one, two or three people to each block of program according to their skills. It is not necessary to assign all of the blocks in the beginning. If possible, assign the Add and Print option first. This will allow you to add and print scores to check both options for proper operation. Later, add Mean to calculate the average of the scores. You will want Change and Delete, which allow you edit and correct the data which have been entered in error. Sort option is used by the Histogram option. Print can be used to check the operation of the Change, Delete and Sort options. The Histogram requires the Sort option so that Sort must be programed before Histogram is tested.

Notes to the team leader: Try not to do the program for the team members, even when they are not programing it the way you would. If the option will work leave the programing to them. Otherwise, you will find yourself doing all of the work which is what you were trying to avoid.

SUMMARY OF WORKSHOP

EVALUATION OF PROCESS

Will the process work for you?
What interfers with the process working for you?
How can the process be optimized for you?

EVALUATION OF PROGRAM

Will the program work for you?
What interfers with the program working for you?
What other things would you want the program to do?

FUTURE EXPANSION OF PROGRAM

What options would you like to add?
What computer will you use?
What resources will you need?
Who can you use as a resource person?

GRADES SOFTWARE SPECIFICATION 15-Jan-82

1. INTRODUCTION

Educators often have to assign grades to a set of test scores. They also have to average grades for a single student. The Grades program will assist the educator in these tasks. First, the Grade program will allow for data entry and editing. Second, it will compute a mean score and histogram. The mean will give the average of a set of scores. The histogram will group the scores and print how many students had each score. The educator will be able to assign grades according to the number of students at each grade.

Basic assumptions about the program.

1. Scores will be entered from keyboard.
2. Only one set of scores will be done at a time.
3. Scores may be sorted.
4. The program will be menu driven:

   When the program starts it will print a list of options and allow you to select the option you want. The program will do the option you select and then return to menu (except when you want to quit).

2. DEFINITIONS and CONVENTIONS

2.1. Terms

The following definitions are applicable for this program.

ARRAY - places where values are stored. Arrays can store a list of values. An example in the program is the array where the scores are stored. in this specification there is one array called a list.

DATA - is defined as integer scores from a test given to a number of students or it is integer scores for a number of scores for a single student.

GLOBAL VARIABLES - are variables that are accessed by different options. Any changes in their value can effect the results of other options.

INTEGER - any whole number, such 5 or 97.

INTEGER VARIABLES - are storage places to store an integer.

LIST - A array where variables are stored.

LOCAL VARIABLES - are variables that are used within the options. Their value is not important outside of the option.

MEAN - is the arithmetical average of the data of the formula: (sum-of-scores)/(number-of-scores).

MENU - is the main part of the program. It will provide a list of the options available to the user. The user will select the options desired and enter a letter indicating his choice. When the options is completed the program will return to the menu and again provide a list of the options available.

HISTOGRAM - is a display of the tally of each value in the scores.

REAL - any real number, which is a whole number and a fraction such as 3.54.

REAL VARIABLES - are places to store a real number.

STRING - a line of text. In a program, strings are usually enclosed in single quotes, such as 'This is a string'.

STRING VARIABLE - is a place to store a string.

VALUE - a number stored in the computer. An example is a test score stored in the computer.

VARIABLE - a place in the computer to store a number or string. It has a name that is usually assigned to it by the programer. The number stored in the variable can be anything the variable was designed for. I. E., whole numbers will fit into integer variables, real numbers will fit into real variables and strings will fit in string variables.

2.2. Notation Conventions

This specification is designed to allow implementation in the programer's choice of language and computer.

Variable names are indicated by the use of hyphens and capital letters such as NUMBER-OF-SCORES. In this specification NUMBER-OF-SCORES variable will store the value of the number of scores which have been entered. It will start at zero and have one added to it each time a score is added.

Values in variables are indicated by the use of pharenthesis around the variable name such as (NUMBER-OF-SCORES). In this specification, when pharenthesis are used around the variable name it means we are interested in the number stored there, such as, add one to (NUMBER-OF-SCORES). In this case we have increased the value stored in NUMBER-OF-SCORES by one because we have added another score to the list of score in the computer. In computer languages references to variables means the values stored in the variable locations.

Print means messages sent to the user. Messages to be printed are enclosed in quotes such as 'Enter data'. Values to be printed as a part of the message are included within the quotes. For example:

'Number of score is (NUMBER-OF-SCORES)'

Input means information entered by user. The information must have a place to be stored, so an input line must also name the location where the input will go, such as, input NUMBER-OF-SCORES. In this instance the uses is asked to enter an integer.

One array is used in this specification. It contains the scores entered and is called LIST-OF-SCORES. Since LIST-OF-SCORES can contain a hundred scores we need a way to indicate which of the scores we want. In this specification the variable, location, of the first score in the LIST-OF-SCORES will be refered to as the First of LIST-OF-SCORES, the value will be expressed as (first of LIST-OF-SCORES). However, in most cases we will be using a variable to indicate which score we want. For example, the last score entered is stored in location (NUMBER-OF-SCORES)th of LIST-OF-SCORES. That is the number stored in NUMBER-OF-SCORES is used to point to the location in LIST-OF-SCORES where the last score is stored. If we wanted to print the last score we would say:

print 'Last score ((NUMBER-OF-SCORE)th of LIST-OF-SCORES)'

3. REQUIREMENTS

3.1. Number of scores

The program will store a minimum of 100 scores. The scores must be stored in a list, an array.

3.2. Menu driven

The program will be menu driven. That is the list of options will be presented to the user at the beginning of the program and at the end of each option, except for the quit option which exits the program.

3.3. Options

Options are:

3.3.1. Add - enters scores into computer. Prompts for each score will be the position in the list where it will be stored. User will be able to return to the menu when the desired scores have been entered. Error exit: The add option will automatically return to menu when maximum number of allowable scores are entered.

3.3.2. Change - replaces a score entered in error. User will enter the score's position in the list and the change option will prompt for the replacement score by giving the position and the current score stored there. Change option will return to menu when replacement score has been entered. Error exit: If there are no scores in the list the change option will return to the menu with the message that there is no data in the list.

3.3.3. Delete - used to delete unwanted scores from the list. User will enter the score's position in the list and the delete option will delete the score. The number-of-scores will be reduced by one. Delete option will return to menu when the deletion is completed. Error exit: If there are no scores in the list the delete option will return to the menu with the message that there is no data in the list.

3.3.4. Mean - computes and outputs the mean of the scores in the list. Option will print the mean and return to the menu. Error exit: If there are no scores in the list the mean option will return to the menu with the message that there is no data in the list.

3.3.5. Print - prints scores and their position in the list. Print option returns to menu when completed. Error exit: If there are no scores in the list the delete option will return to the menu with the message that there is no data in the list.

3.3.6. Sort - sorts scores from low score to high score. Sort option is an option from the menu and histogram will use the sort option. The sort option must return to the menu when used by the menu and return to the histogram option when used by the histogram option. Error exit: If there are no scores in the list the delete option will return to the menu with the message that there is no data in the list.

3.3.7. Histogram - prints a histogram of the scores. Histogram option first sorts the scores then prints the histogram and returns to the menu. Error exit: If there are no scores in the list the delete option will return to the menu with the message that there is no data in the list.

The output format will be:

```
Score  30  Tally   1
Score  60  Tally   3
Score  70  Tally   9
Score  80  Tally  15
Score  90  Tally   6
Score 100  Tally   1
```

3.3.8. Quit - exits the program.

4. SPECIFICATIONS

4.1. Implementation schedule

The menu is the main part of the program, therefore it will be implemented first. The options will be implemented in the following order: Quit, Add, Print, Change, Delete, Mean, Sort, Histogram.

Future implementation will include the saving and retrieving of the scores from a file.

4.2. Program Specifications

4.2.1. Global variables -

LIST-OF-SCORES is an array of 100 integers to store the list of scores.

NUMBER-OF-SCORES is an integer variable to store the number of scores entered.

4.2.2. Menu and Program driver

4.2.2.1. Local variable: CURRENT-OPTION is an string variable to store the selected option.

4.2.2.2. Subprogram Discription:

```
     move zero to (NUMBER-OF-SCORES)

     loop which will -
     print
'Add, Change, Delete, Print, Sort scores, Histogram, Mean, Quit
     input (CURRENT-OPTION)
     select and exercise the appropriate option
     exit the loop when Quit is called
     go to the beginning of the loop

     Quit option.
```

4.2.3. Add option -

4.2.3.1. Local variable: TEMP-SCORE is an integer variable to temporarily store the score.

4.2.3.2. Subprogram Discription:

```
print 'Enter 9999 to end entry of scores'

loop which will -
add one to (NUMBER-OF-SCORES)
if (NUMBER-OF-SCORES) is 101
  then print 'Score list is full'
       subtract one from (NUMBER-OF-SCORES)
       and exit loop
  else Print 'Item (NUMBER-OF-SCORES) -
       input (TEMP-SCORE)
       if (TEMP-SCORE) is 9999
         then subtract one from
              (NUMBER-OF-SCORES)
              exit loop
         else move (TEMP-SCORE) to
               ((NUMBER-OF-SCORES)th
               of LIST-OF-SCORES)
              go to the beginning of the loop
```

4.2.4. Change option -

4.2.4.1. Local variable: POSITION is an integer variable to temporarily store the position in the list that is to be changed.

4.2.4.2. Subprogram Discription:

```
if (NUMBER-OF-SCORES) is 0
  then print 'No scores in list'
       exit option
  else print 'Change which score '
       input (POSITION)
       if (POSITION) is not within the range
               of the current list of scores
         then print
              'Number is outside of the range 1
                     to (NUMBER-OF-SCORES)'
              exit option
         else print 'Score numbered (POSITION)
               is ((POSITION)th of LIST-OF-SCORES)
               change to '
              input ((POSITION)th of
                     LIST-OF-SCORES)
              exit option
```

4.2.5. Delete option -

4.2.5.1. Local variable: POSITION is an integer variable to store the position to be deleted.

4.2.5.2. Subprogram Discription:

```
if (NUMBER-OF-SCORES) is 0
  then print 'No scores in list'
       exit option
  else print 'Delete which score'
       input (POSITION)
       if (POSITION) is not within the range
              of the current list of scores
         then print 'Number is outside of the
              range 1 to (NUMBER-OF-SCORES)'
              exit option
         else move ((NUMBER-OF-SCORES)th of
              LIST-OF-SCORES) to ((POSITION)th
              of LIST-OF-SCORES)
              subtract one from (NUMBER-OF-SCORES)
              exit option
```

4.2.6. Histogram option -

4.2.6.1. Local variables: POSITION,TALLY are two integer variables. POSITON is used to count through the list of scores. TALLY keeps track of the number if identical scores.

4.2.6.2. Subprogram Discription:

```
do the sort option first

if (NUMBER-OF-SCORES) is 0
```

```
        then print 'No scores in list
             exit option
        else move one to (TALLY)
             move one to (POSITION)

             loop which will -
             if ((POSITION)th of LIST-OF-SCORES) is
                    equal to ((POSITION)+1 th of
                    LIST-OF-SCORES)
               then add one to (POSITION)
                    add one to (TALLY)
                    if (POSITION) is equal to
                           (NUMBER-OF-SCORES)
                      then exit loop
                      else go to the beginning of
                           the loop
               else print 'Value ((POSITION)th of
                           LIST-OF-SCORES) Tally
                           (TALLY)'
                    move one to (TALLY)
                    add one to (POSITION)
                    if  (POSITION) is equal to
                           (NUMBER-OF-SCORES)
                      then exit loop
                      else go to the beginning of
                           the loop

           print 'Value ((POSITION)th of
            LIST-OF-SCORES) Tally (TALLY)
```

4.2.7. Mean option -

4.2.7.1. Local variables: POSITION is an integer variable to count through the list of scores. AVERAGE is a real variable where all the scores will be summed and the mean will be calculated.

4.2.7.2. Subprogram Discription:

```
if (NUMBER-OF-SCORES) is 0
  then print 'No scores in list
       exit option
  else move zero to (AVERAGE)
       move one to (POSITION)

       loop which will -
       add ((POSITION)th of LIST-OF-SCORES)
         to (AVERAGE)
       add one to (POSITION)
       if (POSITION) is greater than
              (NUMBER-OF-SCORES)
         then exit loop
         else go to the beginning of loop
```

```
          divide (AVERAGE)/(NUMBER-OF-SCORES),
            put the results in (AVERAGE)
          print 'Mean is (AVERAGE)'
```

4.2.8. Print option -

4.2.8.1. Local variable: POSITION is an integer variable to temporarily store the position in the list that is to be printed.

4.2.8.2. Subprogram Discription:

```
if (NUMBER-OF-SCORES) is 0
  then print 'No scores in list'
       exit option
  else move one to (POSITION)

       loop which will -
       print 'Score (POSITION) is ((POSITION)th
         of LIST-OF-SCORES)'
       add one to (POSITION)
       if (POSITION) is larger than
              (NUMBER-OF-SCORES)
         then exit loop
         else go to the beginning of loop
```

4.2.9. Sort option -

4.2.9.1. Local variables: POSITION,POINTER,TEMPORARY are three integer variables. POSITION is used to count through the list of scores. POINTER is used to point the score which will be used in comparision. TEMPORARY is to be used as a temporary storage of a score.

4.2.9.2. Subprogram Discription:

```
if (NUMBER-OF-SCORES) is 0
  then print 'No scores in list'
       exit option
  else move one to (POSITION)

       loop-A which will -
       move (POSITION) to (POINTER)

       loop-B which will -
       add one to (POINTER)
       if ((POINTER)th of LIST-OF-SCORES) is
            smaller than
            ((POSITION)th of LIST-OF-SCORES)
         then move ((POINTER)th of
                LIST-OF-SCORES) to (TEMPORARY)
              move ((POSITION)th of LIST-OF-SCORES)
         to ((POINTER)th of LIST-OF-SCORES)
       move (TEMPORARY) to ((POSITION)th
         of LIST-OF-SCORES)
  else add one to (POINTER)
       if (POINTER) is larger than
              (NUMBER-OF-SCORES)
         then exit loop-B
         else go to beginning of loop-B

add one to (POSITION)
if (POSITION) is equal to
       (NUMBER-OF-SCORES)
  then exit loop-A
  else go to beginning of loop-A
```

4.2.10. Quit option - When quit option is called it will exit the program and return to the system level.

```
GRADES BASIC PROGRAM 15-Jan-82

100 DIM A(100)
110 N=0
200 PRINT 'A(dd, C(hange, D(elete, P(rint, S(ort data,';
205 PRINT ' Histogram, M(ean, Q(uit';
210 INPUT Q$
220 IF Q$='A' THEN  GOSUB 400
230 IF Q$='a' THEN  GOSUB 400
240 IF Q$='C' THEN  GOSUB 600
250 IF Q$='c' THEN  GOSUB 600
260 IF Q$='D' THEN  GOSUB 800
270 IF Q$='d' THEN  GOSUB 800
280 IF Q$='P' THEN  GOSUB 1000
290 IF Q$='p' THEN  GOSUB 1000
300 IF Q$='S' THEN  GOSUB 1200
310 IF Q$='s' THEN  GOSUB 1200
320 IF Q$='H' THEN  GOSUB 1400
330 IF Q$='h' THEN  GOSUB 1400
340 IF Q$='M' THEN  GOSUB 1600
350 IF Q$='m' THEN  GOSUB 1600
360 IF Q$='Q' THEN  GOSUB 1800
370 IF Q$='q' THEN  GOSUB 1800
380 GO TO 200
400 REM  *****  ADD NEW DATA
410 PRINT 'Enter 9999 to end data entry'
420 N=N+1
430 PRINT 'Item ';N;'- ';
440 INPUT T
450 IF T=9999 THEN 480
460 A(N)=T
470 GO TO 420
480 N=N-1
490 RETURN
600 REM  *****  CHANGE A DATA VALUE
610 IF N>0 THEN 640
620 PRINT 'No data in list'
630 GO TO 720
640 PRINT 'Change which data element';
650 INPUT T
660 IF T<1 THEN 710
670 IF T>N THEN 710
680 PRINT 'Data numbered ';T;' is ';A(T);' change to ';
690 INPUT A(T)
700 GO TO 720
710 PRINT 'Number is outside of the range 1 to ';N
720 RETURN
800 REM  *****  DELETE A DATA ELEMENT
810 IF N>0 THEN 640
820 PRINT 'No data in list'
830 GO TO 920
840 PRINT 'Delete which data element';
850 INPUT T
860 IF T<0 THEN 910
870 IF T>N THEN 910
880 A(T)=A(N)
```

GRADES BASIC PROGRAM 15-Jan-82

```
890 N=N-1
900 GO TO 920
910 PRINT 'Number is outside of the range 1 to ';N
920 RETURN
1000 REM  *****  PRINT DATA
1010 IF N>0 THEN 1040
1020 PRINT 'No data in list'
1030 GO TO 1070
1040 FOR I=1 TO N
1050 PRINT 'Item ';I;A(I)
1060 NEXT I
1070 RETURN
1200 REM  *****  SORT DATA ARRAY
1210 IF N>0 THEN 1240
1220 PRINT 'No data in list'
1230 GO TO 1320
1240 FOR I=1 TO N-1
1250 FOR J=I TO N
1260 IF A(I)<=A(J) THEN 1300
1270 T=A(I)
1280 A(I)=A(J)
1290 A(J)=T
1300 NEXT J
1310 NEXT I
1320 RETURN
1400 REM  *****  PRINT A HISTOGRAM OF THE SORTED DATA
1410 IF N>0 THEN 1440
1420 PRINT 'No data in list'
1430 GO TO 1540
1440 GOSUB 1200
1450 T=1
1460 FOR I=1 TO N-1
1470 IF A(I)<>A(I+1) THEN 1500
1480 T=T+1
1490 GO TO 1520
1500 PRINT 'Value ';A(I);' Tally ';T
1510 T=1
1520 NEXT I
1530 PRINT 'Value ';A(N);' Tally ';T
1540 RETURN
1600 REM  *****  PRINT MEAN OF DATA
1610 IF N>0 THEN 1640
1620 PRINT 'No data in list'
1630 GO TO 1690
1640 T=0
1650 FOR I=1 TO N
1660 T=T+A(I)
1670 NEXT I
1680 PRINT 'Mean ';T/N
1690 RETURN
1800 REM  *****  QUIT THE PROGRAM
1810 END
```

GRADES FORTRAN PROGRAM 15-Jan-82

```
      program grades
      integer list(100),noscor
      common list,noscor
      character curopt

      noscor=0
  100 write(0,200)
  200 format(' A(dd, C(hange, D(elete, P(rint, S(ort data,',\)
      write(0,300)
  300 format(' M(ean, H(istogram, Q(uit ',\)
      read(0,400)curopt
  400 format(a1)
      if ((curopt .eq. 'a') .or. (curopt .eq. 'A')) call add
      if ((curopt .eq. 'c') .or. (curopt .eq. 'C')) call change
      if ((curopt .eq. 'd') .or. (curopt .eq. 'D')) call delete
      if ((curopt .eq. 'm') .or. (curopt .eq. 'M')) call mean
      if ((curopt .eq. 'p') .or. (curopt .eq. 'P')) call print
      if ((curopt .eq. 's') .or. (curopt .eq. 'S')) call sort
      if ((curopt .eq. 'h') .or. (curopt .eq. 'H')) call histog
      if ((curopt .eq. 'q') .or. (curopt .eq. 'Q')) stop
      go to 100
      end

      subroutine add
      integer list(100),noscor
      common list,noscor
      integer tmpscr

  100 write(0,200)
  200 format(' Enter 9999 to end data entry')
  300 noscor=noscor+1
      write(0,400)noscor
  400 format(' Item ',i3,'- '\)
      read(0,500)tmpscr
  500 format(bn,i5)
      if(tmpscr-9999)600,700,600
  600 list(noscor)=tmpscr
      go to 300
  700 noscor=noscor-1
  800 return
      end

      subroutine change
      integer list(100),noscor
      common list,noscor
      integer temp

      if(noscor)100,100,300
  100 write(0,200)
  200 format(' No data in the list')
      go to 1300
  300 write(0,400)
  400 format(' Enter number of data you want to change ',\)
      read(0,500)temp
  500 format(bn,i3)
      if(noscor)600,600,800
  600 write(0,700)
  700 format(' No data in list')
      go to 1300
  800 if((temp .gt. 0) .and. (temp .le. noscor)) go to 1000
      write(0,900)noscor
  900 format(' Number is outside of range 1,',i3)
      go to 1300
 1000 write(0,1100)temp,list(temp)
 1100 format(' Data numbered ',i3,' is ',i5,' change to ',\);
      read(0,1200)list(temp)
 1200 format(bn,i5)
 1300 return
      end

      subroutine delete
      integer list(100),noscor
      common list,noscor
      integer temp

      if(noscor)100,100,300
  100 write(0,200)
  200 format(' No data in the list')
      go to 800
  300 write(0,400)
  400 format(' Delete which data element '\)
      read(0,500)temp
  500 format(bn,i5)
      if((temp .gt. 0) .and. (temp .le. noscor)) go to 700
      write(0,600)noscor
  600 format(' Number is outside of range 1,',i3)
      go to 800
  700 list(temp)=list(noscor)
      noscor=noscor-1
  800 return
      end

      subroutine mean
      integer list(100),noscor
      common list,noscor
      integer local
      real total

      if(noscor)100,100,300
  100 write(0,200)
  200 format(' No data in the list')
      go to 600
  300 total=0
      do 400, local=1,noscor
      total=total+list(local)
  400 continue
      total=total/noscor
      write(0,500)total
 500  format(' Mean ',F8.2)
 600  return
      end

      subroutine print
      integer list(100),noscor
      common list,noscor
      integer local

      if(noscor)100,100,300
 100  write(0,200)
 200  format(' No data in the list')
      go to 600
 300  do 500, local=1,noscor
      write(0,400)local,list(local)
 400  format(' Item ',i3,i8)
 500  continue
 600  return
      end

      subroutine sort
      integer list(100),noscor
      common list,noscor
      integer count1,count2,temp

      if(noscor)100,100,300
 100  write(0,200)
 200  format(' No data in the list')
      go to 500
 300  do 400,count1=1,noscor-1
      do 400,count2=count1,noscor
      if(list(count1) .le. list(count2)) go to 400
      temp=list(count1)
      list(count1)=list(count2)
      list(count2)=temp
 400  continue
 500  return
      end

      subroutine histog
      integer list(100),noscor
      common list,noscor
      integer count,tally

      if(noscor)100,100,300
 100  write(0,200)
 200  format(' No data in the list')
      go to 900
 300  call sort
      tally=1
      do 600,count=1,noscor-1
      if(list(count) .ne. list(count+1)) go to 400
      tally=tally+1
      go to 600
```

GRADES FORTRAN PROGRAM 15-Jan-82

```
400  write(0,500)list(count),tally
500  format(' Value ',i8,' Tally ',i5)
     tally=1
600  continue
     write(0,800)list(count),tally
800  format(' Value ',i8,' Tally ',i5)
900  return
     end
```

273

GRADES COBOL PROGRAM 15-Jan-82

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID.  GRADES.

    ENVIRONMENT DIVISION.
    CONFIGURATION SECTION.
    SOURCE-COMPUTER.  VAX-11.
    OBJECT-COMPUTER.  VAX-11.

    DATA DIVISION.
    WORKING-STORAGE SECTION.
    01  COUNTER-I                   PIC  999 VALUE ZERO.
    01  COUNTER-J                   PIC  999 VALUE ZERO.
    01  TEMPORARY                   PIC  99999 VALUE ZERO.
    01  NUMBER-OF-SCORES            PIC  999 VALUE ZERO.
    01  GRADE-AVE                   PIC  99.9 VALUE ZERO.
    01  CURRENT-OPTION              PIC  X VALUE SPACES.
    01  TABLE1.
        02  TEMP-SCORE              PIC 999.
        02  LIST-OF-SCORES
            OCCURES 100 TIMES       PIC 999.

    PROCEDURE DIVISION.

    MENU.
        PERFORM DECISION-PARA
            UNTIL CURRENT-OPTION = "Q" OR CURRENT-OPTION = "q".
        STOP RUN.

    DECISION-PARA.
        DISPLAY "A(dd, C(hange, D(elete, P(rint, S(ort data,",
            " H(istogram, M(ean, Q(uit".
        ACCEPT CURRENT-OPTION.
        IF CURRENT-OPTION = "A" OR CURRENT-OPTION = "a"
            PERFORM ADD-PARA.
        IF CURRENT-OPTION = "C" OR CURRENT-OPTION = "c"
            PERFORM CHANGE-PARA.
        IF CURRENT-OPTION = "D" OR CURRENT-OPTION = "d"
            PERFORM DELETE-PARA.
        IF CURRENT-OPTION = "H" OR CURRENT-OPTION = "h"
            PERFORM HISTOGRAM-PARA.
        IF CURRENT-OPTION = "M" OR CURRENT-OPTION = "m"
            PERFORM MEAN-PARA.
        IF CURRENT-OPTION = "P" OR CURRENT-OPTION = "p"
            PERFORM PRINT-PARA.
        IF CURRENT-OPTION = "S" OR CURRENT-OPTION = "s"
            PERFORM SORT-PARA.

    ADD-PARA.
        DISPLAY "Enter score to be added.  999 to exit.".
        MOVE 0 TO TEMP-SCORE.
        PERFORM ADD-2
            UNTIL NUMBER-OF-SCORES = 100
            OR TEMP-SCORE = 999.
        IF NUMBER-OF-SCORES = 100
        DISPLAY "Score list is full".

ADD-2.
     ADD 1 TO NUMBER-OF-SCORES.
     DISPLAY "Item ",NUMBER-OF-SCORES.
     ACCEPT TEMP-SCORE.
     IF TEMP-SCORE = 999
         SUBTRACT 1 FROM NUMBER-OF-SCORES
       ELSE
         MOVE TEMP-SCORE TO LIST-OF-SCORES(NUMBER-OF-SCORES).

CHANGE-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE PERFORM CHANGE-SCORE.

CHANGE-SCORE.
    DISPLAY "Chanse which score".
    ACCEPT COUNTER-I.
    IF COUNTER-I < 1 OR COUNTER-I > NUMBER-OF-SCORES
        DISPLAY "Numoer is outside of the ranse ",
          "1 to ",NUMBER-OF-SCORES
      ELSE
        DISPLAY "Score numbered ",COUNTER-I," is ",
          LIST-OF-SCORES(COUNTER-I)," chanse to "
        ACCEPT LIST-OF-SCORES(COUNTER-I).

DELETE-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE PERFORM DELETE-SCORE.

DELETE-SCORE.
    DISPLAY "Delete which score".
    ACCEPT COUNTER-I.
    IF COUNTER-I < 1 OR COUNTER-I > NUMBER-OF-SCORES
        DISPLAY "Number is outside of the ranse ",
          "1 to ",NUMBER-OF-SCORES
      ELSE
        MOVE LIST-OF-SCORES(NUMBER-OF-SCORES) TO
             LIST-OF-SCORES(COUNTER-I)
        SUBTRACT 1 FROM NUMBER-OF-SCORES
        DISPLAY "Score numbered ",COUNTER-I," is deleted".

HISTOGRAM-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE
        PERFORM SORT-PARA
        MOVE 1 TO TEMPORARY
        MOVE 1 TO COUNTER-I
        PERFORM PRINT-HISTOGRAM
          UNTIL COUNTER-I = NUMBER-OF-SCORES
        DISPLAY "Score ",LIST-OF-SCORES(COUNTER-I),
          " Tally ",TEMPORARY.

PRINT-HISTORGRAM.
    ADD 1 COUNTER-I GIVING COUNTER-J.
    IF LIST-OF-SCORES(COUNTER-I) = LIST-OF-SCORES(COUNTER-J)
        ADD 1 TO COUNTER-I
        ADD 1 TO TEMPORARY
      ELSE
        DISPLAY "Score ",LIST-OF-SCORES(COUNTER-I),
         " Tally ",TEMPORARY
        MOVE 1 TO TEMPORARY
        ADD 1 TO COUNTER-I.

MEAN-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE
        MOVE 0 TO TEMPORARY
        PERFORM TOTAL-SCORES VARYING COUNTER-I FROM 1 BY 1
          UNTIL COUNTER-I > NUMBER-OF-SCORES
        DIVIDE TEMPORARY BY NUMBER-OF-SCORES
          GIVING GRADE-AVE ROUNDED
        DISPLAY "Mean of scores ",GRADE-AVE.

TOTAL-SCORES.
    ADD TEMPORARY LIST-OF-SCORES(COUNTER-I) GIVING TEMPORARY.

PRINT-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE
        PERFORM PRINT-ALL
        VARYING COUNTER-I FROM 1 BY 1
        UNTIL COUNTER-I > NUMBER-OF-SCORES.

PRINT-ALL.
    IF COUNTER-I = 1 DISPLAY "Number      Score".
    DISPLAY COUNTER-I,"           ",LIST-OF-SCORES(COUNTER-I).

SORT-PARA.
    IF NUMBER-OF-SCORES = 0
        DISPLAY "No scores in list"
      ELSE
        PERFORM SORT-I
        VARYING COUNTER-I FROM 1 BY 1
        UNTIL COUNTER-I = NUMBER-OF-SCORES.

SORT-I.
    PERFORM SORT-J
    VARYING COUNTER-J FROM COUNTER-I BY 1
    UNTIL COUNTER-J > NUMBER-OF-SCORES.

SORT-J.
    IF LIST-OF-SCORES(COUNTER-I) > LIST-OF-SCORES(COUNTER-J)
```

```
           MOVE LIST-OF-SCORES(COUNTER-I) TO TEMP-SCORE
           MOVE LIST-OF-SCORES(COUNTER-J) TO LIST-OF-SCORES(COUNTER-I)
           MOVE TEMP-SCORE TO LIST-OF-SCORES(COUNTER-J).
```

GRADES PASCAL PROGRAM 15-Jan-82

```
program grades;

var current_option:char;
    number_of_scores:integer;
    list_of_scores:array[1..100] of integer;

     procedure add;

     var temp_score:integer;

     begin {add}
       temp_score:=0;
       writeln('Enter 9999 to end data entry');
       repeat
         begin
           number_of_scores:=number_of_scores+1;
           write('Item ',number_of_scores:3,'- ');
           readln(temp_score);
           if temp_score<>9999
             then list_of_scores[number_of_scores]:=temp_score
             else number_of_scores:=number_of_scores-1
         end
       until (temp_score=9999) or (number_of_scores=100);
       if number_of_scores=100
         then writeln('Data list is full')
     end {add};

     procedure change;

     var position:integer;

     begin {change}
       if (number_of_scores<1)
         then writeln('No data in list')
         else
           begin
             write('Enter number of data you want to change ');
             readln(position);
             if not (position in [1..number_of_scores])
               then writeln('Number is outside of range [1..'
                             ,number_of_scores,']')
               else
                 begin
                   write('Data numbered ',position,' is '
                          ,list_of_scores[position],' change to ');
                   readln(list_of_scores[position])
                 end
           end;
     end {change};

     procedure delete;

     var position:integer;
```

```
begin {delete}
 if (number_of_scores<1)
  then writeln('No data in list')
  else
   begin
    write('Delete which data element');
    readln(position);
    if not (position in [1..number_of_scores])
     then writeln('Number is outside of range [1..'
                ,number_of_scores,']')
     else
      begin
       list_of_scores[position]
        :=list_of_scores[number_of_scores];
       number_of_scores:=number_of_scores-1
      end {else}
   end {else}
end {delete};

procedure sort;

var position,pointer,temporary:integer;

begin {sort}
 if (number_of_scores<1)
  then writeln('No data in list')
  else
   begin
    for position:=1 to number_of_scores-1 do
    for pointer:=position+1 to number_of_scores do
     if list_of_scores[position]>list_of_scores[pointer]
      then
       begin
        temporary:=list_of_scores[position];
        list_of_scores[position]:=list_of_scores[pointer];
        list_of_scores[pointer]:=temporary
       end {if};
   end {else}
end {sort};

procedure histogram;

var position,tally:integer;

begin {histogram}
 sort;
 if (number_of_scores<1)
  then writeln('No data in list')
  else
   begin
    tally:=1;
    for position:=1 to number_of_scores-1 do
     if list_of_scores[position]=list_of_scores[position+1]
      then
```

```
                   begin
                     tally:=tally+1;
                     if position+1=number_of_scores
                       then writeln('Value ',list_of_scores[position]:8,
                                    ' Tally ',tally:5)
                   end
                 else
                   begin
                     writeln('Value ',list_of_scores[position]:8,
                             ' Tally ',tally:5);
                     if position+1=number_of_scores
                       then writeln('Value ',
                                    list_of_scores[number_of_scores]:8,
                                    ' Tally ',1:5);
                     tally:=1
                   end
           end
     end {histogram};

     procedure mean;

     var position:integer;
         total:real;

     begin {mean}
       if number_of_scores<1
         then writeln('No data in list')
         else
           begin
             total:=0.0;
             for position:=1 to number_of_scores do
               total:=total+list_of_scores[position];
             writeln('Mean ',total/number_of_scores:8:2)
           end;
     end {mean};

     procedure print;

     var position:integer;

     begin {print}
       if number_of_scores<1
         then writeln('No data in list')
         else
           for position:=1 to number_of_scores do
             writeln('Item ',position:3,list_of_scores[position]:8);
     end {print};

begin {grades}
  number_of_scores:=0;
  current_option:='~';
  while not (current_option in ['q','Q']) do
    begin
      write('A(dd, C(hange, D(elete, P(rint, S(ort data, M(ean, ');
      write('H(istogram, Q(uit ');
      read(current_option);
      writeln;
      case current_option of
        'a','A':  add;
        'c','C':  change;
        'd','D':  delete;
        'm','M':  mean;
        'p','P':  print;
        's','S':  sort;
        'h','H':  histogram;
        'q','Q':  ;
      end {case};
    end {while}
end {grades}.
```

HOME-BREWED SOFTWARE

by

Richard C. Dixon, MBA
C.E.R.F. Inc.
1441 N. 27th Lane
Phoenix, Arizona 85009

In the past, when I have recommended "home-brewed" software to someone, I'm often asked if that is not just re-inventing the wheel? My answer is a definite maybe! It really depends upon the original inventor of this particular wheel. In order to form a basis from which to discuss the "professional programmer" vs. the "novice", I think it is necessary to look briefly at the professional.

Just like rare wines, programmers come in different vintages, but it will suit our purposes here to separate them into just two; early and late.

In the 1950's, computers were hardly more than a twinkle in some scientist's eye and most of the research being done was in the universities. Those computers have since become known as first generation computers. The programmers were the inventors, engineers, scientists, and mathematicians who also were generally involved with the design and manufacture of their electronic brains.

By the early '60's, the second generation computer was here and many a visionary professional made a sudden career change and jumped on the computerized bandwagon of professional programmers. Some of my very good programmer friends are of this vintage. These programmers brought with them a variety of experiences into the field. They cut their teeth on front panels with flashing lights, machine language programming, and launched the infant computer industry into the '70's.

The typical late vintage programmer, on the other hand, is a young person who goes directly from high school into some type of computer training in a college or trade school. He is taught four or five high-level languages, flow-charting, decision charts, and how to make a computer do tricks and sing songs. His other training and experiences are meager by any standard, and therefore he must rely upon a systems analyst for direction. Incidently, most systems analysts are prior programmers who have been promoted.

If this trend continues, I can envision a time when the blind will be leading the blind; programs will run at incredible speeds; displays will look like something in an arcade; printouts will be formatted and printed to perfection; and the huge amounts of information generated will be absolutely worthless.

When should the wheel be re-invented? I certainly wouldn't recommend an educator try and develop a computerized reading program if he can buy a good one at his local computer store, unless of course he is a confirmed masochist. On the other hand, if there is not a program on the market, he only has two choices. He can either wait for someone else to write a program or write one himself. If he waits, someone knowledgeable in teaching reading skills may write a program in which case everything will be fine. If a "professional programmer" with very little knowledge of teaching reading skills writes the program, then there may be a lot of "bells and whistles" and very little real learning potential.

If you do not accept the points I have made so far as valid then I apologize for wasting your time. if you agree with at least some of the points, then I urge you to continue reading on how to develop "home-brewed" software.

1. Thinking the Problem Through. The first thing is to eliminate any preconceived ideas you may have concerning flow-charts, decision tables, etc., and get back to something more familiar like outlining.. Visualize or even relate the events that must transpire within a particular project. Write a one line "descriptor" for each event and then proceed to the next event. You may need to do this step several times to insure the events are all inclusive. After all the events are recorded, arrange them in the order they must be performed in. I find it easier to number them before trying to re-arrange them.

SEE EXAMPLE ON FOLLOWING PAGE

Example: Here would be the descriptors for a simple program to calculate the average gasoline cost for an automobile.

② Input distance travelled
① Clear Screen
③ Input amount of gasoline used (in gallons)
④ Input the price per gallon of gasoline
⑤ Calculate average miles per gallon
⑥ Calculate total cost for trip
⑨ Display average cost per mile
⑦ Calculate cost per mile
⑩ Display average miles per gallon
⑪ Display total cost for trip
⑧ Clear screen

As can clearly be seen, this program could be extended and enhanced to include such things as the computer calculating the gasoline costs for a certain distance and/or calculating the other automobile expenses, but this should be sufficient to illustrate the use of descriptors as a programming aid.

2. Expanding or Condensing the Program. As the theme must fit the scope of any writing endeavor, so must the constraints be fitted to a computer program. One problem with programs in general is they're too restrictive. Another problem is they're too comprehensive. For example, if a payroll program is written for ten employees in only one catagory then it will be much too restrictive. The first time another employee is hired or another catagory is added the program will have to be re-written. Vice versa, provisions for 400 employees and 100 catagories may be too comprehensive. The set-up and operating time required may negate any benefit gained from computerizing the system. With this in mind, establish the constraints needed in your program, ie., the number of students, words, problems, etc., and list these under constraints.

3. Defining the Variables. A variable is something that may change when the program is run as opposed to a constant which will always remain the same. Most people are already familiar with the concept of variables when used for numbers in mathematics. In addition, the programmer uses a second type of variable which is called

a "string variable" . String variables are alpha-numerical data that changes, such as names, addresses, the names of months, etc.

Have you ever worked on an algebraic problem only to discover when you were finished that you had also forgotten what the variables you used were supposed to represent? If you have, then you know it's a frustrating experience and you probably only used two or three variables. In a computer program, you will be using many more variables, perhaps 50 times as many, so you can see it is important to make a variable list.

Even though you may have the variables in a list it is bothersome to continually refer to the list each time you want to use a variable. Computers have in this respect made the programmers life a little more bearable. Depending upon the computer language being used, a full word may be used for a variable name instead of just a single letter. Most computers that use the BASIC language will allow you to use a full word as a variable name, but the computer recognizes only the first two letters. If you should use two words with the same first two letters, like PAy and PAyment you will be in trouble.

In the example used earlier with the descriptors, the variable list may look like the one below:

Variable List

MILES = Distance travelled
GAL = Amount of gasoline used
PRICE= Price per gallon
AVERAGE= Average cost per mile
MPG= Average miles per gallon
COST= Total cost of trip

Notice that there are no duplications in the first two letter of any of the variables. Using these variables the formulas would be:

MPG=MILES ÷ GAL
COST= GAL X PRICE
AVERAGE= COST ÷ MILES

Now you should be ready to talk to the electronic brain about your problem.

4. Coding. If you tried to order a meal in a French restaurant and did not understand the french language you probably wouldn't get much (take my word for it, I tried it). Trying to communicate with a computer without knowing the language will get you even less. You must learn the computer's language. I'll guarantee it won't learn yours even if you are an American. I suggest you take an introductory programming class at a local school if your not already on speaking terms with your computer. Next, get access to a computer that uses the language you want to use (you can purchase a micro with six languages for less than $2000). LEARN! ENJOY! HAVE FUN! You don't have to lose your "novice" standing to code your program into the computer.

5. Debugging and Polishing. If your program is now up and running and doing pretty much what you designed it to do, you may now want to polish it a bit. In other words, add a few of those "bells and whistles" I was so down on earlier.

Debugging is not entirely just getting rid of those nasty syntax errors and diagnostics. It is also making sure all the output is correct, that entering certain data, even things that should not be entered, does not cause a "crash" and for providing "operator comfort" ie., keeping the operator informed on what the computer is doing. When your program is running fairly error free, you may now want to visit a software consultant such as the one I work for in Phoenix, C.E.R.F. Inc. (Computers in Education and Related Fields). They may be able to offer suggestions on how to increase speed and efficiency or how to improve screen formating and/or input routines, etc.

Remember though, if you do go to a consultant, listen to his suggestions, but in the final analysis, it is your program. You have the expertise in the field in which your writing, so don't let a consultant take away some of the good qualities of your program.

It has been said in speaking of women, that "there is no such thing as a 10," well I don't know about that, I see quite a few that rate up there close, but I DO know there are NO perfect computer programs. After some time you will probably want to make some changes in your original program. All I can say is have at it, and you'll be thankful for the documentation you put together earlier.

6. Documentation. There are several ways to document a program once it is written. Much of it has already been done with the descriptors, constraints, and variables lists. In addition, there are at least two other things you should provide for documentation. First is a program listing, which is simply a list of the instructions you coded into the computer. This is very easy to obtain if you have a printer, but is often overlooked and omitted. The second item is a procedural or operating manual. One way to write the narrative for this manual is to simply run the program and record each prompt in the program with the proper response and/or action needed. Include any unique characteristics and/or data requirements. Do not forget to include a backup copy of your program diskette/tape along with the other documentation.

7. Protection and Marketing. After you have spent a few weeks or months creating your masterpiece what do you do with it except use it? Of course your primary purpose in writing the program was to get the use of it, but if you needed a program that wasn't available and your job is not a unique, one-of-a-kind then there must be others who could benefit from your program. If you're the benevolent type, you may want to make the program "public domain." If so, possible users will still have to be made aware of its existence, but this can be done through organizations, like computer user's groups, educational associations, or state departments. If you should decide on this method of disseminating your program, you won't have to worry about protecting it, but who will bear the cost of reproducing the program and the mailing costs? Additionally, will the end user be able to get the full benefit from the program without an operating manual? And, if a manual is provided who will publish it and how much will be charged for it? I think most of you would agree that someone giving something away would not want to provide these additional services.

A second alternative is to put your program in the hands of a software company. There are two currently used methods of doing this. First, you can sell the program out-right for a certain sum of money. In this case you relinquish all rights to the program with the exception that you would probably still be listed as the author. The other method allows you to retain a right in the program and be paid a royalty for each copy sold. In either way, the software company would reproduce the program, publish operating instructions, and perform the marketing operations for your program. The software company would want some software protection on the program to protect their investment. The protections they use may include some or all of the following items. The company's programmers would modify some of your program using machine language to make it difficult to "break" or list the program. They may put some essential parts of the program in ROM (read only memory) which consists of pre-programming a chip (this is usually called "firmware") which is then inserted into the computers circuit. They may instead use a key device which connects into one of the computer's user ports. They may also copyright the program and/or have a licensing agreement with end-users or dealers. Of course there is no ultimate software protection and software protection is more to keep the "honest people honest" and to discourage "unauthorized users" than to deter professional code breakers.

Summary. Now if you have your project in mind, let's go through the steps once again:

1. Outline the project.
2. List the constraints.
3. List the variables.
4. Code the program.
5. Debug and Polish the program.
6. Document the program.
7. Protect and Market the program.

Don't expect to get rich making "home-brewed" software, but then again I have assumed your expertise is in some other field. Writing a program will give you a wonderful sense of pride and accomplishment when you see it run. It may also in part take the place of that book you always wanted to write, but which most of us never get around to, and last of all, it may provide the funds for you to buy that other program you've always wanted and hoped you would never have to write!!

PROGRAMMING FOR HIGH SCHOOL STUDENTS

A typical "math-computer teaching project" for highschool students:

TO FIND THE FIRST DERIVATIVE OF A POLYNOMIAL FUNCTION OF DEGREE N AT A GIVEN VALUE OF X = A USING SYNTHETIC DIVISION.

Theory: Let F(X) be a polynomial function of degree N. Then $\frac{F(X)}{X - A} = G(X) + R$. A polynomial function divided by a linear factor is a polynomial function of degree N - 1 plus a remainder R.

Thus $F(X) = (X - A) \cdot G(X) + R$, the first derivative of F is $F'(X) = (X - A) \cdot G'(X) + G(X) \cdot 1$ and $F'(A) = 0 \cdot G'(A) + G(A) = \underline{G(A)}$

Example: Find F'(5) given $F(X) = 3X^3 - 2X^2 + X - 10$
divide F(X) by X - 5 using synthetic division

$$\overset{F(X)}{\frac{3X^3 - 2X^2 + X - 10}{X - 5}} = \overset{G(X)}{3X^2 + 13X + 66} + \overset{R}{\frac{320}{X - 5}}$$

$$\begin{array}{rrrrr} 5) & 3 & -2 & 1 & -10 \\ & & 15 & 65 & 330 \\ \hline & 3 & 13 & 66 & \boxed{320} \end{array}$$

Now find G(5) by the remainder theorem (synthetic division)

$G(X) = 3X^2 + 13X + 66$

$G(5) = 206$

Thus $F'(5) = G(5) = 206$

$$\begin{array}{rrrr} 5) & 3 & 13 & 66 \\ & & 15 & 140 \\ \hline & 3 & 28 & \boxed{206} \end{array}$$

```
 10  REM  FIND FIRST DERIVATIVE OF A FUNCTION BY SYNTHETIC DIVISION
 20  G(0) = 0: H(0) = 0
 30  PRINT "ENTER THE DEGREE OF YOUR FUNCTION": INPUT N: PRINT: PRINT
 40  PRINT "ENTER THE COEFICIENTS OF THE POLYNOMIAL IN DESENDING
     POWER ORDER.  IF A TERM IS MISSING, ENTER ZERO": PRINT: PRINT
 50  FOR I = 1 TO N + 1: PRINT "COEFICIENT #";I;: INPUT A(I): NEXT I
 60  IF N = 0 THEN GOTO 120: PRINT: PRINT
 70  PRINT "FOR WHAT VALUE OF X DO YOU WANT THE FIRST DERIVATIVE?;:
     INPUT A: PRINT
 80  REM  DO SYNTHETIC DIVISION NOW:
 90  FOR I = 1 TO N + 1: G(I) = A(I) + A * G(I - 1): NEXT I
100  FOR I = 1 TO N: H(I) = G(I) + A * H(I - 1): NEXT I
110  PRINT " THE FIRST DERIVATIVE AT X = ";A;" IS "; H(N)
120  END
```

by Robert Zenor
Math Department
Coronado High School
2501 N. 74th Street
Scottsdale, AZ 85257

VII. GLOSSARY AND JOURNALS

GLOSSARY

ADDRESS — A number that designates a location where information is stored in a memory device.

ALGORITHM — A statement of mathematical and/or logical steps to be followed in the solution of a problem.

ASCII — *A*merican *S*tandard *C*ode for *I*nformation *I*nterchange. A standard set of binary codes which represent letters, numbers and symbols.

ASSEMBLER — A program which converts an assembly language (usually English mnemonics) into a machine language.

ASSEMBLY LANGUAGE — A computer language that uses "shorthand" forms to represent one or more machine language instructions.

BASIC — *B*eginners *A*ll-Purpose *S*ymbolic *I*nstruction *C*ode. A high-level conversational programming language in wide use. Incorporates simple English words and common mathematical symbols.

BATCH PROCESSING — A technique of submitting program(s) to a computer in "bunches." The computer processes programs one after the other, with no provision for user input. Very efficient use of computer time. Contrasts with *Interactive Processing*.

BAUD — A rate of data transfer, given in bits per second. Alphabetic characters usually require about 10 bits per character, so a Baud rate of 300 corresponds to about 30 characters per second.

BINARY CODE (DIGIT) — The most fundamental of codes, using only 0's and 1's to represent data. Can be represented by the presence or absence of electrical current within key parts of a computer.

BIT — A contraction for *Bi*nary Digi*t*.

BOOLEAN — A system of logic design which was invented by George Boole. Utilizes simple functions such as AND, OR, NOT, NOR, etc.

BUG — Something which prevents a program from working properly. Usually an error in typing or program design, although sometimes it can be a malfunction within the computer itself.

BUS — The set of electrical lines connecting the various parts of the computer. Data and control signals are sent along these lines.

BYTE — A basic unit of information in a computer. A byte usually represents one character and is normally eight bits in length.

CAI — *C*omputer *A*ssisted *I*nstruction. A means to set the computer to interact as a teacher on a particular topic.

CHARACTER — Letters, numbers and symbols which can be arranged into information. A character is usually defined with one byte of data.

CHARACTER SET — Those characters available to any particular computer or peripheral device. Some devices have a limited character set; for example, some printers can print only capital letters.

CHIP — A rectangular silicon die cut from a "wafer". It is a logic unit of the computer.

CLOCK — The internal pace-setter of a computer. The clock keeps a number of events within the computer working together at an even pace.

COMPILER — A program which translates a high level language into a machine-readable program. Normally allows the computer to execute a program much more rapidly.

COMPUTER — A general purpose machine for computing that contains a CPU, memory, input/output devices, cables and cabinet. The machine does many tasks otherwise done by hand.

CPU — *C*entral *P*rocessing *U*nit. The "brain" of the computer. Directs all the functions of the computer.

CRT — *C*athode *R*ay *T*ube. The tube which makes the television screen. Used to display text and a graphics created by a computer. Frequently linked with a keyboard to create a CRT terminal.

CURSOR — A movable spot on the face of a CRT which indicates where the next character will be displayed.

DATA — Information transferred to or from a computer.

DATABASE — A systematic way of storing data files for later processing.

DEBUG — The process of locating and eliminating errors.

DISC (DISK) — A mass storage device capable of high storage and retrieval speeds. Similar in appearance to a magnetic oxide coated record. Can be either floppy or hard, depending upon performance requirements.

DISKETTE — A floppy disc, a circular mylar colated disc that is used to store digital information on for later use. Can be programs and/or data.

DOS — *D*isc *O*perating *S*ystem. A collection of programs which combine with an interface to allow a computer to control a disc machine.

DOT MATRIX — A technique of using an array of dots to create characters. A 5 x 7 array of dots is the bare minimum for alphanumeric characters, while larger arrays, such as 7 x 9 will allow upper and lower case letters, and underlining.

DUPLEX — An interface which allows simultaneous two-way communication between a computer and a peripheral device.

EPROM — *E*rasable *P*rogrammable *R*ead *O*nly *M*emory. A long term memory which, under special conditions, can be erased and rewritten.

EXECUTION — The actual running of a computer program.

EXTERNAL STORAGE — (Mass Memory) Data storage through a peripheral device, such as a tape or disc machine.

FILE — A set of data which has some specified relationship.

FIRMWARE — Programs which are stored in ROM. They are immediately available for execution. There is no need to load them from a storage device.

FLOPPY DISC — A mass storage device that is flexible like a soft record, yet stores a large amount of data or programs.

FLOWCHART — A diagrammed flow of program logic using symbols to show the logic of the program to see if they contain the things the programmer desires of the program.

FORTRAN — *For*mula *Tran*slator. A high level language oriented toward math and science applications.

GIGO — *G*arbage *i*n, *g*arbage *o*ut. Any errors which enter a computer will result in errors in the output.

GRAPHICS — The ability of the computer to construct line drawings, graphs, charts, etc. on a CRT or printer.

HALF DUPLEX — An interface which allows either reception or transmission of data, but does not allow both to occur simultaneously.

HARDCOPY — Computer output which is printed onto paper. Contrasts with the temporary image presented on a CRT.

HARDWARE — The equipment which makes up a computer system.

HEXADECIMAL SYSTEM — A number system based upon 16. Uses the digits 0-9 and letters A-F.

HIGH-LEVEL LANGUAGE — A computer language with characteristics of English words, decimal arithmetic, and common mathematical symbols. Each instruction usually represents many individual computer operations.

IC — *I*ntegrated *C*ircuit. Miniaturized electronic devices which have many distinct components integrated into self-contained circuits. Responsible for the reduction of room sized computers into microcomputers.

INPUT — Information (or data) entering a computer or peripheral device. The same information may be output from some other part of the system.

INSTRUCTION — A single directive to the computer; a command to do something.

INTEGRATED CIRCUIT — A circuit put on a chip of silicon.

INTERACTIVE — A computer which maintains communication with the user during operation. Allows the use of the computer for instruction.

INTERFACE — A hardware and/or software device used to connect a computer to peripheral equipment.

INTERPRETER — A program which translates a language (usually into machine code) as it is being executed.

I&O — *I*nput/*O*utput of information. The two terms are frequently used together because they often involve the same communication lines.

K OR KILO — Symbol or prefix for 1,000. In computer language, 1 K actually stands for 2 to the tenth power, or 1,024.

KEYBOARD — A collection of keys and buttons similar to a typewriter with added keys to control the functions of a terminal and/or computer. Information is entered into the computer directly through a keyboard.

LANGUAGE — The method by which one communicates with a computer. Languages vary from machine level codes through high level types, such as BASIC.

LARGE SCALE INTEGRATION (LSI) — The technology which enables thousands of semiconductor devices to be etched onto one silicon chip to do a multitude of operations.

LIGHT PEN — An input device for the CRT screen that allows quick entry of common data by touching a screen with this device at certain points. The certain points are all preprogrammed to do certain things at each point.

LINE PRINTER — A printer capable of printing whole lines of data at one impact of the printer per line. 40 to 132 characters at a time can be printed in this fashion. Makes for high speed printing capabilities.

LOAD — The process of inserting information (a program or data) into a computer.

LOOP — A group of instructions that can be performed repeatedly until certain conditions are met.

MACHINE LANGUAGE — A language which can be executed directly by the computer. Usually a series of binary or hexadecimal numbers. Very difficult for humans to work with.

MAIN STORAGE — Directly assessable space in computer memory for programs and the data stored in memory locations for high speed operation.

MASS STORAGE — Devices such as discs or tapes which can store large quantities of program and data. These programs and data must be loaded into the computer's RAM and ROM before they can be used.

MEMORY — The high speed electronic components in a computer which store information called RAM and ROM before they can be used.

MICROCOMPUTER — A very small computer, usually desk-top in size. Presently, microcomputers can process only 8 or 16 bits of information in any one step.

MICROPROCESSOR — An application of the LSI chip for a complete processor which includes the arithmetic logic unit and the control unit.

MNEMONIC — Intended to assist human memory. Usually abbreviations or related alphabetic characters. Examples: dvd for divide, sub for subtract, etc.

MODEM — *Mo*dulator-*Dem*odulator. An interface which allows computers to communicate over telephone lines.

MOTHER BOARD — The circuit board in the microcomputer which interconnects the CPU with all other memory and support devices.

ON-LINE — Directly connected to the computer.

OPERATING SYSTEM — Software required to run the computer hardware. It keeps the programs and related hardware going in the timing and other factors required to make a computer function.

OUTPUT — Information or data coming from a computer.

PARALLEL COMMUNICATION — A technique for the I&O of information which allows bytes (8 bits) to be transmitted simultaneously. Very fast.

PASCAL — A relatively new high level language. Named in honor of the French mathematician.

PERIPHERAL DEVICE — Equipment which links to a computer. Printers, CRT's, tape and disc drives are examples.

PROGRAM — A series of instructions to a computer. User types in the way certain things are to be done by the computer to generate the final results in "reports" or other user oriented uses.

PROM — *P*rogrammable *R*ead *O*nly *M*emory. Permanently recorded data stored on special RAM memory IC's. Once programmed, they can't be altered by computer or man. Useful for storing frequently utilized instructions.

RAM — *R*andom *A*ccess *M*emory. The "working" memory of the computer. This memory can be accessed and altered by the computer as needed. This is a form of memory which is very susceptible to power interruptions or program "bugs".

RANDOM ACCESS — Accessing data based on knowledge of the address to locate the data.

ROM — *R*ead *O*nly *M*emory. About the same as PROM, except that ROM is usually programmed at the time of manufacture. Can't be altered by the computer or man.

RS232C — A standard of voltage levels used for the serial input/output of data. Commonly called RS-232.

RUN — A command used in many languages to start the execution of a program.

SCROLLING — The movement of the contents of a line on a screen up or down one at a time giving the lines the appearance of moving up or down.

SOFTWARE — Programs and necessary documentation which are needed to make a computer "do its thing."

SYSTEM — The complete computer setup including the CPU, memory, input/output devices, plus any other devices required to run the applications wanted on the computer.

TERMINAL — A peripheral device which usually consists of a printer or a CRT and a keyboard.

TIME SHARING SYSTEM — A means to have many input areas to one main computer with each user sharing time with the other. The computer operating system makes sure all users on line can access the system, usually without much noticeable delay.

VIDEO DISC — An optical system in which images and sound are captured on a recordlike disc. When the disc is played on a video player, a laser beam "reads" the "information" on the disc and translates it into video signals that are reproduced on a TV set connected to the video player.

VIDEOCASSETTE TAPES — Cartridgelike cassette tapes which are slipped into a videotape player that is hooked up to a TV set for display.

Association of Media Producers

COMPUTER JOURNALS

MICROCOMPUTER LABORATORY
COLLEGE OF EDUCATION
ARIZONA STATE UNIVERSITY

AEDS Journal
1201 16th Street N.W.
Washington, D.C. 20036

AEDS Monitor
St. Paul Public School
360 Colborne St.
St. Paul, MN 55102

BYTE
70 Main St.
Peterborough, NH 03458

Classroom Computer News
P.O. Box 266
Cambridge, MA 02138

COMPUTE Magazine
P.O. Box 5406
Greensboro, NC 27403

Computer Using Educators
1776 Education Park Drive
Independence High School
San Jose, CA 95133

Computers in Mathematical
Science Education
014 Memorial Hall
Newark, DE 19711

Computers in Mathematics
and Science Teaching
P.O. Box 4825
Austin, TX 78765

The Computing Teacher
Computing Center
Eastern Oregon State College
La Grande, OR 97850

CourseWare Magazine
4919 N. Millbrook #222
Fresno, CA 93726

Creative Computing
P.O. Box 789-M
Morristown, NJ 07960

Datamation
666 Fifth Avenue
New York, NY 10103

Educational Computer
P.O. Box 535
Cupertino, CA 95015

Electronic Education
Suite 220
1311 Executive Center Drive
Tallahassee, FL 32301

Electronic Learning
Scholastic Inc.
50 W. 44th Street
New York, NY 10036

Infoworld
530 Lytton Avenue
Palo Alto, CA 94301

Interface Age
McPheters, Wolfe and Jones
16704 Marquardt Avenue
Box 1234
Cerritos, CA 90701

Microcomputing
Elm Street
Peterborough, NH 03458

Personal Computing
1050 Commonwealth Avenue
Boston, MA 02215

Popular Computing
70 Main Street
Peterborough, NH 03458

Recreational Computing
People's Computer Co.
1263 El Camino Peak, Box E
Menlo Park, CA 95025

Robert Purzer's Magazine
P.O. Box 466
El Dorado, CA 95623

T.H.E. Journal
Synergy, Inc.
P.O. Box 992
Acton, MA 01720
